UNIVERSIDAD CARLOS III DE MADRID
Departamento de Ingeniería Telemática

Proyecto Fin de Carrera

HyperAuthor: Una herramienta para la creación de narrativa hipertextual

Autor: Diana Espinal Cruces
Tutor: Dr. José Jesús García Rueda

Leganés, Junio de 2009

PROYECTO FIN DE CARRERA

Departamento de Ingeniería Telemática

Universidad Carlos III de Madrid

Título: HyperAuthor: una herramienta para la creación de narrativa hipertextual

Autor: Diana Espinal Cruces

Tutor: Dr. José Jesús García Rueda

EL TRIBUNAL

Presidente: **Maria Blanca Ibáñez Espiga**

Secretario: **Ricardo Romeral Ortega**

Vocal: **Domingo Sánchez Mesa Martínez**

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 18 de Junio de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Presidente Secretario Vocal

Resumen: HyperAuthor consiste en una herramienta de creación de ficción hipertextual cuyo principal objetivo no consiste únicamente en facilitar al autor la escritura de esta clase de literatura, sino también en producir un tipo de hipertexto narrativo que sea capaz de adoptar las principales características de esta nueva forma de escribir sin perder aquellos importantes logros de los libros e historias tradicionales, alcanzados después de muchos siglos de mejora. HyperAuthor es, por lo tanto, una herramienta pensada para crear hiperficción teniendo siempre presente que, en el momento actual, alejarse demasiado de la "literatura convencional" podría causar una pérdida de calidad en la escritura y la lectura. El objetivo reside incorporar algunas características del hipertexto que ayuden a reforzar la relación íntima que se establece entre lector y relato, evitando al mismo tiempo los obstáculos que estas nuevas características puedan provocar. En este sentido, HyperAuthor representa sólo un pequeño paso adelante desde el libro tradicional que, por un lado, ayudaría a incrementar el número de lectores que consumen este nuevo tipo de literatura, y por el otro, reduciría el tiempo que estos lectores necesitan para adoptar la narración de historias interactiva como algo propio y natural.

Agradecimientos

Quiero dar las gracias a todos aquellos quienes, para bien o para mal, voluntariamente y sobre todo sin saberlo, me han dado ánimos. Gracias a vosotros este proyecto por fin ha visto la luz.

Gracias a mi padre, un buen hombre, que me ha enseñado mucho. Gracias a mi madre, una mujer buena, que desde siempre y aún hoy, cuida de mí. Gracias a los dos por todo durante tantos años y, sobre todo, por no perder nunca la confianza en mí. Sin vosotros, no sería lo que soy.

Gracias a mi hermana (no me olvido de ti), por nada en concreto y por todo a la vez. Por toda una vida juntas, que no es poco, y por toda una vida por delante para compartir.

Gracias a mis familiares, que me soportan y han compartido conmigo este largo camino.

Gracias a mi principito, por hacerme sentir la más especial de entre todas las rosas. Gracias por todos los preciosos momentos que me ha regalado. Porque está a mi lado, me escucha y al final se ríe conmigo. Qué sería de la vida sin él.

Gracias a mis amigos, los de verdad, los de toda la vida, los amigos con mayúsculas. Aquellos que, sin necesidad de nombrarlos, saben que a ellos están dirigidas estas palabras. Gracias por dejarme formar parte de su vida y llenar la mía, día tras día, de momentos inolvidables.

Y finalmente, no puedo dejar de mencionar a mi tutor. Gracias a José Jesús, que me corrige y me enseña mucho. Gracias por encontrar siempre la manera de indicarme el mejor camino para continuar.

Índice

Pliego I
Memoria

Capítulo 1: Introducción

1.1 Contexto

Desde hace algún tiempo, y cada vez con más frecuencia, los estudiosos de la literatura se detienen a reflexionar sobre el futuro que deparará a la misma esta nueva era en que vivimos, una era rebosante de información e incesantes avances tecnológicos. En la actualidad son muchos los libros que se publican acerca de las complicadas relaciones que se están estableciendo entre las Humanidades y las nuevas tecnologías, acerca del futuro del libro, de su posible desaparición o de su supervivencia, de la forma en la que la literatura sabrá adaptarse y aprovecharse de las nuevas formas de comunicación.

En este contexto surge el hipertexto, una nueva forma de estructurar información textual capaz de superar las limitaciones de la página impresa. Si la imprenta supuso en su momento una revolución en los conceptos de documento, autor, lector, edición, información y conocimiento, el hipertexto supone una nueva vuelta de tuerca que hace necesario redefinir estos conceptos. Si la tecnología determina las formas de pensamiento y expresión, es presumible que la llegada de esta nueva tecnología dé lugar a nuevas formas culturales.

Pero cabe preguntarse si es realmente posible hacer llegar esta nueva forma de creación a un público cuyo principal interés es el ocio. Y parece que la respuesta es que sí. En un mundo audiovisual donde la televisión o los videojuegos tienen una mayor aceptación que la lectura, las características de inmersión, interactividad, etc. que presenta la literatura hipertextual hacen que ésta deba tener una gran aceptación entre un público que reclama dinamismo e interactividad.

Se debe, por lo tanto, crear productos literarios que sean capaces de ofrecer al lector la posibilidad de interacción "manual" como complemento de la interacción "intelectual" típica de las obras tradicionales. Productos en los que el lector sea capaz de influir en el desarrollo de la historia, sin olvidar que el componente literario de calidad debe estar siempre presente.

En nuestra opinión, las posibilidades de este nuevo género en una sociedad como la actual son grandes y están aún sin explorar. Decidimos, por lo tanto, construir una aplicación que tuviera en cuenta los principios anteriormente mencionados y que permitiera crear narrativa hipertextual de forma sencilla. La herramienta implementada recibe el nombre de HyperAuthor, y junto con ella se ha desarrollado también un visor para los hipertextos creados mediante su uso.

Pero veamos los motivos por los cuales decidimos abordar este proyecto.

1.2 Motivación

En la actualidad, la hiperficción debe ser todavía considerada como una forma de arte "joven", que se encuentra aún en sus inicios. Las obras hipertextuales existentes en la actualidad no son capaces de aprovechar al máximo todo el potencial que el hipertexto posee y ésta es una situación cuya duración en el tiempo es indeterminada. La creación literaria no se ajusta con facilidad a un proceso de evolución detallado y planificado, y es por esto por lo que puede que haya que esperar aún durante un largo periodo de tiempo antes de que la época de madurez de este nuevo género sea una realidad.

Ante esta situación se puede optar por esperar a que la narrativa hipertextual evolucione hasta alcanzar una madurez similar al que ha alcanzado la literatura tradicional, o se puede empezar a experimentar con ella para crear productos con los que el público se familiarice permitiendo, de esta manera, acelerar su evolución.

El desarrollo de este proyecto se fundamenta en la creencia en la segunda de las opciones mencionadas, apoyando esta opinión en la postura de autores como Howard S. Becker, quien en su artículo "*Ficción hipertextual, una nueva forma de arte*" (ver [3]) desarrolla el concepto "mundo de arte". Un mundo de arte para Becker consiste en la red de actividades cooperativas que involucra a todas las personas que contribuyen para que la obra de arte se realice, se conozca y se comparta. Igualmente, la existencia y valoración de obras de arte dependen de la retroalimentación que su "mundo" haga de ellas, y la hiperficción, entendida como una nueva forma del arte de narrar, no puede ser menos.

Por este motivo es importante que esta nueva forma de literatura pueda empezar a llegar al gran público, a pesar de que los intentos actuales de aprovechar las ventajas de la tecnología para enriquecer la literatura no estén aún a la altura de los logros alcanzados por la literatura tradicional. Lo más importante en este momento en el que se encuentra la hiperficción no es crear obras maestras, sino popularizar los productos de la narrativa hipertextual.

De esta manera, es posible experimentar un desarrollo mucho más veloz. Como se afirma en [9] "*si el género se populariza entre los lectores, entonces llamará la atención de más escritores, que crearán más obras, aumentando la oferta y atrayendo más a los lectores. Y según aumente el número de escritores y de obras, mayores y más rápidos avances tendrá el género*".

El desarrollo de un mundo viable de ficción hipertextual debe superar el mundo pequeño y restringido de algunos pocos lectores y escritores, para extenderse a una gran masa de público, de manera que la nueva forma de arte encuentre eco y posibilidades de desarrollo. Afirmar que la ficción hipertextual es un arte nuevo implica no sólo reconocer la novedad estética intrínseca, sino, desde un punto de vista sociológico, la posibilidad práctica de que los recursos y realizaciones necesarios para que esta forma artística penetre en el público se den realmente.

Y aún es posible llegar más lejos ya que, en la actualidad, las formas más populares de ocio tienen una componente narrativa más o menos fuerte: la literatura, el cine y el teatro

son claros ejemplos, pero también los videojuegos y los chats tienen una componente de "quiero saber que va a pasar a continuación", además de incluir una componente nueva de "quiero influir en lo que va a pasar a continuación". Y estos son los dos puntos sobre los que se basa la narrativa hipertextual. Por este motivo, hoy más que nunca, se puede decir que la narrativa puede abarcarlo todo, todo es narrativa.

En este punto del análisis, parece interesante esbozar algunas directrices sobre cómo debería abordarse el problema de popularizar esta nueva forma de arte. Existen algunos aspectos cuya inclusión en las nuevas obras las hace más atrayentes para el público actual. Estos aspectos a tener en cuenta según [9] son:

- Multimedia: Sin llegar a transformarse en productos típicamente audiovisuales, sí que los usuarios de las actuales tecnologías de la información esperan encontrar algo más que la palabra escrita.

- Interacción fuerte: La pasividad no está permitida en los nuevos medios. El diálogo constante con el dispositivo que se está manejando debe ser afianzado con la interacción. Esto marca la diferencia entre la narrativa hipermediática y otras formas narrativas estrictamente multimediáticas, como el cine.

- Entornos colaborativos y comunidades virtuales: El establecimiento de lazos de comunicación con otros usuarios es necesario para que los usuarios de las redes no se sientan aislados.

A modo de resumen se puede concluir que las necesidades identificadas para la popularización de este nuevo género son: organizaciones hipertextuales, elementos multimedia y una gran componente interactiva. Sin embargo, hay algo de notable importancia que no se debe omitir en este recuento: las herramientas tecnológicas necesarias para que obras con las características anteriormente mencionadas puedan ser una realidad.

Por este motivo, y con el fin de facilitar el correcto diseño y elaboración de hipertextos, las herramientas específicas para este género son cada vez más numerosas en el mercado. Sin embargo, la mayoría de las herramientas existentes en la actualidad presentan una serie de limitaciones que impiden que se estén aprovechando todas las ventajas que el hipertexto puede de ofrecer. Muchas de las obras que se encuentran actualmente en Internet y que pretenden ser narrativa hipertextual no son más que versiones mejoradas de las notas al pie de página, resultando en obras que preservan la mayoría de los elementos de la ficción lineal y no incluyen aquellas innovaciones que la hiperficción puede aportar.

En resumen, el hipertexto es aún un género incipiente con un gran potencial que necesita de la evolución de las herramientas tecnológicas para mostrarse en todo su esplendor. El desarrollo de una herramienta como HyperAuthor pretende dar un

pequeño paso en el proceso de evolución que es necesario recorrer desde la literatura lineal hacia la literatura hipertextual.

En este sentido la herramienta debe ajustarse a un modelo diferente de literatura que introduzca en la literatura los preceptos y ventajas de la hipertextualidad. Entre estas aportaciones se encuentran la interactividad proporcionada por los enlaces o los elementos multimedia como complemento del texto. Sin embargo, este modelo (que se desarrollará con detalle en apartados posteriores), tiene como misión principal la de permitir y facilitar la construcción de hiperficciones que no "vayan demasiado lejos" a fin de no perder los importantes logros alcanzados por la narrativa tradicional, destacando entre ellos el tiempo profundo.

Este tiempo profundo se entiende como la unión que el lector experimenta con la obra que está leyendo y que hace que el lector se sienta unido a la historia, que sienta el relato como algo propio. La hiperficción debe conservar esta característica, al igual que muchas otras de la literatura tradicional, si se quiere alcanzar una hiperficción de calidad. Por lo tanto, cualquier elemento de decisión en el texto debe apoyar el tiempo profundo así como cualquier elemento multimedia debe aumentar la fusión entre relato y lector. Ésta es la línea en la que HyperAuthor pretende aportar su "pequeño granito de arena".

1.3 Descripción de los objetivos

El objetivo de este proyecto consiste en el diseño e implementación de una herramienta software completa que facilite la creación de narrativa hipertextual por parte de los autores. Esta herramienta debe permitir el desarrollo de hiperficción sin eliminar los logros conseguidos por la narrativa tradicional e incorporando algunas características del hipertexto que ayuden a aumentar la fusión entre el relato y el lector.

Las obras hipertextuales están basadas en un enfoque en el cual al usuario se le permite el acceso a los documentos de una manera no secuencial (diferenciándose de la literatura más tradicional en la que el acceso era naturalmente secuencial). Además da al lector la libertad de tomar decisiones sobre sus trayectos de lectura, permitiéndole la elección de los enlaces que quiere visitar en cada momento. Esta flexibilidad de acceso genera las nociones de navegación y personalización de presentaciones, lo que favorece una lectura profunda por parte del lector. La herramienta de autor desarrollada debe favorecer la existencia de esa profundidad en las obras creadas mediante su uso.

En términos generales, la aplicación debe incluir la funcionalidad adecuada para permitir al autor crear hipertexto, organizarlo en estructuras e incluir reglas de navegación. La herramienta debe permitir que el autor decida, con total libertad (aunque apoyándose siempre en estructuras "tipo") sobre la conectividad del documento que está creando, por lo que la herramienta construirá un mapa del documento o mapa conceptual que ayude al autor durante el proceso de creación. Al mismo tiempo, debe incluir una

interfaz de usuario apropiada, lo suficientemente sencilla e intuitiva como para que pueda ser usada por personas sin grandes conocimientos sobre tecnología.

En último lugar, la herramienta debe permitir la visualización del libro electrónico de forma que el autor pueda experimentar sensaciones semejantes a las del lector. Este módulo de visualización ofrecerá al autor la posibilidad de corregir y mejorar su creación a medida que avanza, así como de exportar el documento para su almacenamiento y distribución.

De forma más concreta, los objetivos en este sentido son:

☑ Diseñar, haciendo hincapié en la usabilidad y la funcionalidad, una interfaz de usuario que no presuponga unos conocimientos técnicos avanzados por parte del usuario.

- Proporcionar una representación icónica adecuada tanto de las estructuras narrativas "tipo" como de la estructura de la obra en concreto.
- Implementar un acceso fácil y rápido a todas las funcionalidades de la aplicación basado en el uso del ratón y las teclas de acceso directo.
- Implementar un método totalmente gráfico de construcción de la estructura de las obras narrativas basado en el uso del ratón.
- Implementar un editor para el contenido de la obra intuitivo y familiar, que permita no sólo la inclusión de texto sino también de imágenes y enlaces.

☑ Conseguir un entorno de trabajo interactivo en el que el usuario pueda crear obras hipertextuales, modificarlas y revisarlas durante su creación, así como visualizarlas de un modo que, aunque diferente al del lector, se asemeje en la medida de lo posible.

☑ Aprovechar las capacidades multiplataforma del lenguaje Java implementando la aplicación en éste lenguaje.

☑ Añadir nuevas funcionalidades a la herramienta creadora mediante la implementación de una herramienta de visualización. Los objetivos de ésta son:

- Proporcionar una herramienta pensada en el lector de narrativa hipertextual y no en el autor de la obra.
- Favorecer un entorno adecuado de lectura, que favorezca la unión entre el lector y el relato (eliminando todos aquellos elementos ajenos al proceso de lectura, p.ej. las barras de herramientas)

- Implementar un mecanismo de navegación por la estructura hipertextual basada en el uso del ratón y en la división de la pantalla de visualización en "zonas".

Por otro lado, hasta el momento se ha hecho hincapié en el hecho de que HyperAuthor nace con el objetivo de favorecer el proceso de maduración de un género aún incipiente como es la hiperficción. En este sentido, y para posibilitar una mejor comprensión de este proceso,se pretende realizar un pequeño análisis de la evolución de la hiperficción desde su aparición hasta nuestros días, así como de las diferentes teorías existentes sobre el "futuro del libro" y el modo en el que la nueva herramienta encaja en este contexto. Del mismo modo también es objetivo de este proyecto extraer conclusiones del análisis anterior y esbozar una situación deseable para la hiperficción en el futuro así como unas primeras pautas para alcanzarla.

Por último, en una segunda fase del proyecto se decidió realizar una evaluación de la herramienta consistente en un pequeño estudio de campo que abordara algunos aspectos técnicos de la misma, tales como su robustez, su estabilidad, etc.

1.4 Introducción a HyperAuthor

Antes de entrar en materia, parece conveniente presentar la aplicación resultante de todo este proceso para que en los siguientes capítulos se disponga de alguna referencia. Resumiendo muy brevemente, HyperAuthor es un entorno de creación de narrativa hipertextual que presenta dos niveles de diseño diferenciados: la dimensión estructural y la dimensión de contenido. La construcción de la estructura de la obra se lleva a cabo mediante un entorno totalmente gráfico y para ello se dispone de un conjunto reducido de estructuras tipo universales.

HyperAuthor es, por lo tanto, una herramienta orientada de una forma clara hacia el autor de hiperficción. Como complemento a HyperAuthor se ha desarrollado de forma paralela HyperViewer, un visor cuyo foco, en este caso, se sitúa en el lector. El propósito consiste en proporcionar una herramienta de lectura que compartiera la filosofía del modelo utilizado como base para el desarrollo de HyperAuthor y en la que se pudieran visualizar de una fácil y cómoda aquellas obras creadas con la herramienta de autor que nos ocupa.

En las siguientes figuras se pueden observar ambas aplicaciones en funcionamiento.

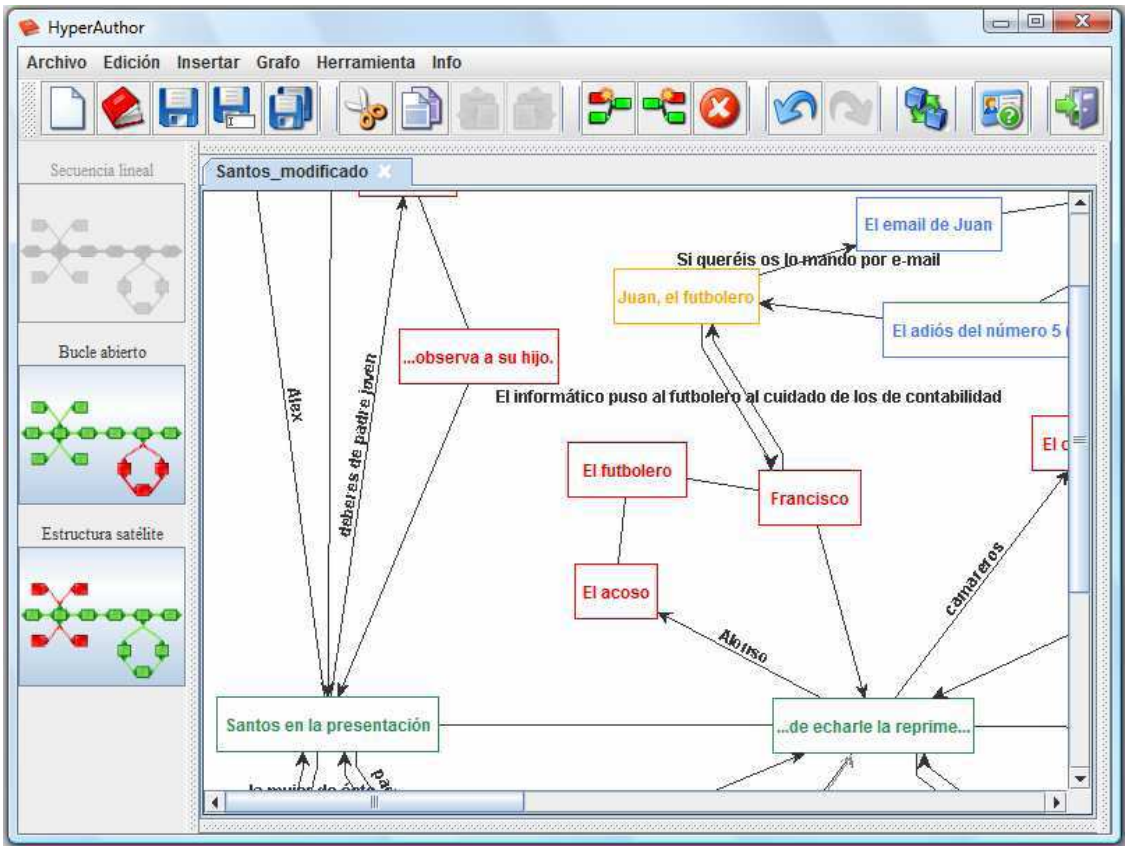


Figura 1.1 HyperAuthor en funcionamiento.

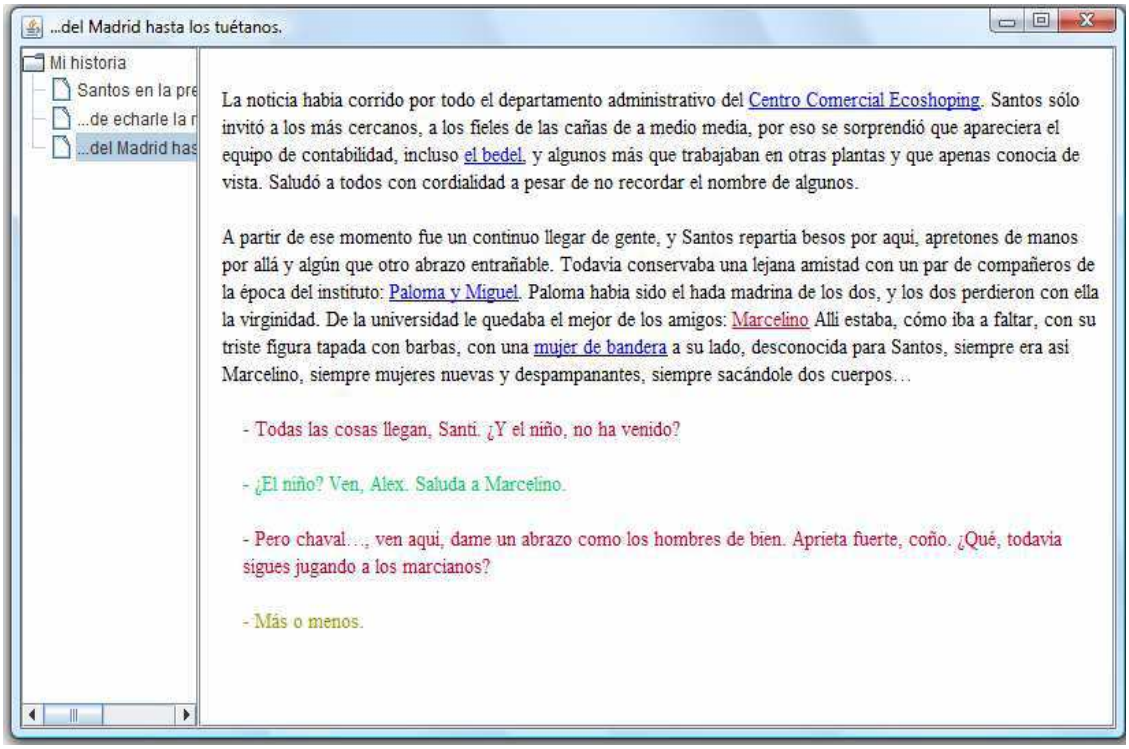


Figura 1.2 HyperViewer en funcionamiento.

Para la creación de HyperAuthor se tomó como base una serie de requisitos identificados durante el análisis de la situación actual de la hiperficción así como de las herramientas que se encuentran a día de hoy en el mercado. Con esta descripción funcional se comenzó el proceso de diseño de la interfaz de usuario, haciendo especial énfasis en su sencillez y usabilidad. Se abordó posteriormente el diseño del núcleo de la herramienta y la asociación de las distintas funcionalidades a la interfaz. En último lugar se implementó HyperViewer, con el fin de visualizar las obras ya creadas con HyperAuthor.

En los capítulos siguientes se describe en detalle tanto el proceso de diseño e implementación de la herramienta como el resto de los puntos del proyecto, entre los que se encuentran los experimentos y el estudio de las distintas teorías sobre la evolución de la narrativa y las conclusiones extraídas. En el siguiente punto se puede encontrar una guía que ayudará al lector a localizar cada uno de estos aspectos a lo largo del documento.

1.5 Estructura del documento

Este apartado constituye una guía del presente documento, por lo que en él se realiza un recorrido por los diferentes capítulos que lo componen:

- Capítulo 1: Introducción. Es el capítulo en el que se encuentra actualmente, y contiene una introducción al resto del documento así como los objetivos y motivaciones del mismo.

- Capítulo 2: Hipertexto y literatura. Constituye un estudio ensayístico sobre las diferentes teorías de pensamiento sobre el futuro del libro, así como el proceso evolutivo llevado a cabo hasta la situación actual. Se esbozan también las primeras directrices sobre lo que no se debe dejar atrás para poder alcanzar la madurez de este nuevo género.

- Capítulo 3: Estado del arte. Encontrará una breve recopilación de trabajos relacionados con HyperAuthor en las distintas líneas en las que se ha desarrollado este proyecto.

- Capítulo 4: Diseño. Describe el proceso de diseño de la aplicación comentando todas aquellas decisiones adoptadas que aportan riqueza a la descripción de la solución.

- Capítulo 5: Implementación. Describe el proceso de desarrollo software de la aplicación a un nivel fundamentalmente técnico, así como las pruebas técnicas llevadas a cabo con el producto terminado.

- Capítulo 6. Conclusiones y trabajos futuros. Incluye las conclusiones obtenidas a partir del trabajo realizado y las líneas en las que se puede continuar este proyecto.

Capítulo 2:
Hipertexto y literatura

2.1 Una nueva forma de expresión

Históricamente, y a causa de las restricciones que presentaba ya en su momento el papiro y actualmente un formato como el de la página impresa, la lectura se ha realizado tradicional y exclusivamente de una forma secuencial.

Esta forma de lectura está tan arraigada en el lector, que éste entiende la linealidad como una de las principales características del texto. El texto impreso es finito, con un comienzo, un final y una historia definida. Como norma general, el papel no permite saltarse los límites de la página impresa y tanto la lectura como la escritura se desarrollan línea a línea de arriba a abajo y siguiendo un orden secuencial dentro de cada página o página a página. Existen algunas excepciones a esta norma en la página impresa, son las llamadas pioneras del hipertexto, como por ejemplo la novela *"Rayuela"* de Julio Cortázar, pero estas excepciones serán analizadas más adelante.

El desarrollo de la tecnología convierte en una realidad aquello que las novelas pioneras del hipertexto habían esbozado con anterioridad. La aparición del hipertexto permite dar un salto cualitativo en un género como es el narrativo, ya que esta nueva tecnología posee una serie de características con un gran potencial para literatura. Ante esta nueva realidad, conviven posturas de signo muy diferente. Por un lado se encuentran aquellos que entiende la hipertextualidad como un cambio tan profundo que transformará todos los aspectos de la vida cotidiana. En el extremo opuesto se hallan aquellos quienes también opinan que el hipertexto representará un cambio radical en las formas de comunicación, pero en esta ocasión piensan que el fin del libro impreso tiene como consecuencia inevitable el fin de la cultura. En último lugar, y en una posición a media distancia entre las dos anteriores se encuentran los escépticos, quienes aceptan la existencia de un nuevo paradigma, el hipertextual, pero al contrario que los demás, no creen que éste implique cambios sustanciales.

Para poder entender la argumentación de cada una de las posturas, es oportuno realizar en este punto del documento un pequeño análisis histórico de las diferentes corrientes de pensamiento que se han dado a lo largo de las últimas décadas.

Los críticos de la narrativa hipertextual se apoyan en la existencia de una corriente de pensamiento que en los últimos 30 años ha ganado mucha importancia gracias a las ideas de personas como Roland Barthes o Michael Foucault.

La noción sobre el texto que Barthes formuló a principios de la década de los 70 era ya muy similar al actual hipertexto. *"Pienso en un texto formado por bloques de palabras(o de imágenes), electrónicamente unidos por múltiples trayectos, cadenas o recorridos dentro de una textualidad abierta, eternamente inacabada y descrita mediante conceptos como nexo, nodo, red, trama y trayecto"*.

Foucault, por su parte, concibe el texto no como algo independiente, sino como algo que, debido a su naturaleza, existe necesariamente en relación con otros textos. En su

obra *"La arqueología del saber"* (ver [41]) afirmaba que *"las fronteras de un libro nunca están claramente definidas, ya que se encuentra atrapado en un sistema de referencias a otros libros, otros textos, otras frases: es un nodo dentro de una red... una red de referencias"*.

Esta corriente de pensamiento ha dado lugar, por un lado, a la existencia de una serie de obras hipertextuales enfocadas hacia la crítica literaria, y por otro, al surgimiento de una nueva corriente narrativa que explora las posibilidades del nuevo medio hipertextual.

Sin duda una figura clave en la aplicación de la tecnología hipertextual al mundo de las Humanidades, y en concreto al de la Literatura, es George P. Landow. En [16], publicado en 1992, Landow afirma que en las décadas anteriores se había ido produciendo, de forma paulatina pero continua, un acercamiento entre dos campos aparentemente sin conexión alguna: la teoría de la literatura y el hipertexto informático. De esta forma relaciona el pensamiento de aquellos que se dedican a la informática, como Nelson o van Dam, con aquellos otros que se dedican a la teoría cultural como Derrida y Barthes. Todos ellos, *"como otros muchos especialistas del hipertexto y teoría cultural, postulan que deben abandonarse los actuales sistemas conceptuales basados en nociones como centro, margen, jerarquía y linealidad y sustituirlos por otras como multilinealidad, nodos, nexos y redes"*.

Sin embargo, el hipertexto también ha contado y cuenta con un gran número de detractores. En la posición contraria a Landow se encuentran otros autores y pensadores que critican los textos fragmentados, discontinuos y sin una línea argumental definida a los que el hipertexto da lugar.

Como se ha podido comprobar, existen corrientes de pensamiento en ambos extremos, tanto a favor como en contra del hipertexto. De cualquier forma, lo verdaderamente importante es que, al igual que cambios como el de lo oral a lo escrito o el nacimiento de la imprenta fueron significativos en su momento y perduran aún en la actualidad, igual ocurrirá con la era electrónica y el hipertexto. La tecnología transforma la sociedad a pasos agigantados en un proceso que afecta a todos los aspectos de la vida, y en lo que se centrará este análisis es en la repercusión de la tecnología hipertextual en la práctica literaria.

2.2 Hipertexto y literatura

El hipertexto consiste en un método de presentación y organización de la información donde determinadas palabras o imágenes en el documento que se visualiza pueden ser expandidas para obtener información adicional referente a esa palabra o imagen dentro del mismo documento o fuera de él. En otras palabras, hipertexto es un texto en el que los datos se almacenan en una red de nodos conectados por enlaces. Los nodos contienen texto, y si además contienen gráficos, imágenes, audio o cualquier otro recurso, recibe el nombre de hipermedia.

Considerando cómo se representa el conocimiento humano, el cerebro opera por asociación, saltando de un objeto al siguiente de forma casi instantánea. El paradigma hipertextual (o el hipermedia) intenta reproducir este proceso con enlaces entre fragmentos de información contenidos en nodos. El proceso de lectura y escritura del hipertexto se caracteriza tanto por su no secuencialidad como por la libertad de movimiento que permite al usuario debido a su estructura; gracias a ella, los usuarios no están obligados a seguir una secuencia establecida.

Esta flexibilidad en la estructura ha favorecido la aplicación de las técnicas hipertextuales en campos muy heterogéneos, lo que ha permitido que el hipertexto se aplique en diferentes materias como la informática, matemáticas, lógica, lingüística, pedagogía, teoría literaria, etc. Una de las principales tendencias ha sido identificar el hipertexto con la literatura. Ya Ted Nelson, uno de los primeros introductores, llamó a los ordenadores "máquinas literarias" y en este mismo sentido las obras ya clásicas de George P. Landow enfocan el hipertexto desde el punto de vista de la crítica literaria.

Como ya se ha esbozado anteriormente, el análisis de la relevancia del hipertexto en la literatura no va a centrarse en si existen o no formas distintas de narrar, ni en dilucidar cuál responde a la forma real de funcionamiento del pensamiento, si al razonamiento lógico tradicional o al razonamiento por asociación. Lo cierto es que la racionalidad humana sigue siendo la misma y las dos son constitutivas del pensamiento. Pensamos secuencial y simultáneamente y también de forma asociativa. Quizá las asociaciones no pueden ser representadas al hilo del discurso pero, en parte, el hipertexto nos las hace presentes, y eso es lo bueno del hipertexto, que permite ambas.

Es importante saber qué aporta la estructura hipertextual, y, sobre todo, destacar que el hipertexto introduce nuevas formas de organizar la información y nuevas formas de acceso a ésta. El hipertexto ofrece también una nueva forma de presentar, acceder y recuperar dicha información, otra cuestión es cómo se estructura el hilo argumental de un documento y si sigue una determinada línea narrativa o argumentativa, varias o incluso ninguna. Lo cierto es que el hipertexto permite todas y cada una de estas líneas narrativas gracias a enlaces de muy distinto tipo: jerárquicos, asociativos, formales, conceptuales, referenciales, explicativos, etc. y que permite diferentes formas de representación de las ideas, la información y el conocimiento, desde una línea estrictamente secuencial, hasta una combinación de relaciones secuenciales, jerárquicas de distinto tipo y asociativas de diferente signo.

El hipertexto, al igual que el libro tradicional, exige un lector activo y reflexivo que esté familiarizado con el nuevo medio y, al igual que se exige que un lector de libros deba estar acostumbrado a la lectura lineal, el usuario de un hipertexto deberá conocer las claves para usar, navegar y explorar un hipertexto. Por su parte, los gráficos, las imágenes en movimiento, los sonidos y todas las posibilidades multimedia conforman toda una riqueza de recursos y posibilidades que nos ofrece, de forma secuencial y simultánea (multisecuencial) el hipertexto.

Todas estas posibilidades que ofrece el hipertexto permiten que éste se haya introducido en la literatura de la mano de tres aproximaciones diferentes, la tercera de ellas de corte bien distinto a las anteriores:

- las llamadas ediciones genéticas, consecuencia de aplicar ideas como la de obra abierta a la crítica hipertextual
- la narrativa hipertextual, también conocida como hiperficción
- las ediciones críticas de obras clásicas o hiperedición

Dejamos al margen de este análisis la existencia de obras impresas que se han adaptado al medio digital y al hipertexto, pues en Internet se puede encontrar un gran número de textos clásicos y modernos que se han adaptado a la escritura hipertextual para poder ser leídos y navegados en línea y que están accesibles en diversas bibliotecas digitales.

Las **ediciones genéticas** surgen de la idea de Borges de la Biblioteca de Babel, un libro en el que caben todos los libros, ya que permite una lectura a la carta, un recorrido por el proceso de síntesis de una obra y un estudio detallado de ésta, además de situar a la obra en contexto con otros textos.

Los primeros hipertextos de este tipo fueron *"In Memoriam"*, creado por Landow y sus alumnos usando el programa *Intermedia*, y *"Forking Paths: An Interaction after Jorge Luis Borges"*, de Stuart Moulthrop construido con *Storyspace*.

En *"In Memoriam"*, al poema de Tennyson, los alumnos añadieron nexos tales como variantes sacadas de manuscritos, críticas publicadas, comentarios de los propios alumnos, pasajes de obras de otros autores, etc. Landow afirma que la obra de Alfred Lord Tennyson está plagada de alusiones y referencias y que se prestaba especialmente bien al tratamiento hipertextual.

Se entiende por **narrativa hipertextual** o **hiperficción**, a una serie de hipertextos que exploran las posibilidades estructurales y formales del nuevo medio y que permite al lector elegir diferentes caminos de lectura alejándose de la linealidad. En estas obras aparece una nueva forma de narrar, y entre ellas abundan tanto las obras de narrativa como la poesía hipertextual. Se trata de literatura experimental que explora las posibilidades de la nueva forma de narrar no secuencial, como alternativa a la narrativa lineal propia de las novelas tradicionales del medio impreso. Es en este tipo de obras en las que se centrará el análisis más adelante.

Por último, la **hiperedición** consiste en la aplicación del hipertexto a las ediciones críticas de las obras clásicas. En el formato libro tradicional, las ediciones críticas suelen ser muy difíciles de leer y su utilización es poco manejable precisamente porque la herramienta utilizada (el libro) es igual a la materia estudiada.

La llamada hiperedición se ha demostrado muy útil puesto que los enlaces facilitan el recorrido por las anotaciones críticas, los glosarios, las variantes de versiones, etc. Todas estas herramientas permiten visualizar y enlazar más allá del primer nivel de lectura por

medio de estructuras complejas, bibliografías descriptivas, formas de referencia abreviadas...

2.3 Narrativa hipertextual o hiperficción

El fenómeno que se va a abordar en esta sección se corresponde con las creaciones literarias no sólo realizadas en un soporte electrónico, sino también pensadas para ser transmitidas y recibidas en ese mismo soporte y que son conocidas con el nombre de hiperficción. En otras palabras, y de acuerdo a Susana Pajares Tosca, experta en hipertexto, se entiende por narrativa hipertextual aquellas obras escritas específicamente para el medio electrónico y no las ediciones hipertextuales de obras escritas para ser publicadas en forma de libro.

La narrativa hipertextual o hiperficción es algo más que el simple hecho de vincular documentos a través de herramientas informáticas. Son necesarias *organizaciones hipertextuales*, elementos multimedia, formas creativas de narración y herramientas que hagan más sencilla la confluencia entre tecnología y literatura.

Existe ya bastante bibliografía sobre esta supuesta nueva forma de creación literaria, que algunos entienden como un género literario nuevo y otros como una nueva forma de experimentación literaria o, más aún, como la única posibilidad que tiene la literatura en la actualidad de ensayar y experimentar nuevas formas de creación. El problema al que se enfrenta este nuevo género es la carencia de integración entre teoría y práctica.

2.3.1 Antecedentes

Como se ha afirmado reiteradamente, no existe hipertexto sin los medios digitales. Sin embargo, existen innumerables ejemplos fuera del ámbito tecnológico de lo que se podrían denominar como precursores del hipertexto. Se trata no sólo de elementos como las citas, referencias bibliográficas o notas al pie de página que desde la antigüedad se pueden encontrar en manuscritos y libros impresos, sino también de obras literarias más recientes que intentan ofrecer al lector distintas lecturas no secuenciales manteniéndose, sin embargo, dentro de los límites de la página impresa.

Entre las obras consideradas como antecesoras de las verdaderas novelas hipertextuales suelen citarse títulos de tanta relevancia como *"Finnegan´s Wake"* de Joyce, *"Pale Fire"* de Nabokob, *"Tristram Shandy"* de Sterne, *"El castillo de los destinos cruzados"* o *"Si una noche de invierno un viajero"*, de Italo Calvino, *"Ficciones"*de Borges o *"Rayuela"* de Julio Cortázar.

En los sucesivos párrafos se realiza una breve descripción de aquellas características más relevantes de algunas de estas obras de manera que quede patente su relación con el paradigma hipertextual.

Comencemos por la que probablemente se considera como la novela precursora del hipertexto por excelencia, *"Rayuela"*. Uno de los aportes de esta novela radica en que la forma en que se acostumbra a leer constituye únicamente una de las infinitas formas en las que se puede leer e interpretar esta novela. El propio Cortázar propone dos: *"A su manera, este libro es muchos libros, pero sobre todo, es dos libros"*. El primero de estos dos libros se lee de manera tradicional, comenzando en la primera página y siguiendo un orden lineal hasta que se alcanza el capítulo 56, donde finaliza. El segundo de los libros se lee siguiendo las instrucciones del "tablero de dirección" que aparece en las primeras páginas del libro.

En lo relativo a Borges, son muchas las obras en su haber que destacan como precursoras de la ficción hipertextual. El ejemplo más representativo es quizá *"El jardín de los senderos que se bifurcan"*, una novela de la que el autor dice *"el tiempo se bifurca perpetuamente hacia innumerables futuros"*; bifurcación es sinónimo de hipertexto. En ella se puede leer *"Ts'ui Pen diría una vez: me retiro a escribir un libro. Y otra: me retiro a construir un laberinto. Todos imaginaron dos obras; nadie pensó que libro y laberinto eran un solo objeto."*

"La Biblioteca de Babel" es otro de los ejemplos. Borges imagina una biblioteca que abarca todos los libros, una biblioteca ilimitada y periódica en la que cada libro es todos los libros. *"El Universo (que otros llaman la Biblioteca) se compone de un número indefinido, y tal vez infinito, de galerías hexagonales. (...) No me parece inverosímil que en algún anaquel del universo haya un libro total. (...) Yo me atrevo a insinuar una solución del antiguo problema: La Biblioteca es ilimitada y periódica."*

Por último, y para mencionar alguna obra no escrita en lengua castellana, *"Vida y opiniones del caballero Tristram Shandy"*, narra la vida de un personaje del siglo XVII pero entrelazando historias e incluyendo fragmentos de obras de otros autores. Por su parte, Italo Calvino también escribió obras que preconizaban la aparición del hipertexto, a las que él mismo denominó "hipernovelas". Por ejemplo, en *"El castillo de los destinos cruzados"*, Calvino utiliza las cartas del Tarot para contar 12 historias que se entrecruzan y que conforman el argumento de la novela. En *"Si una noche de invierno un viajero"*, Calvino organiza la obra como una serie de relatos entrelazados a modo de hipertexto. El protagonista de la obra descubre, debido a una serie consecutiva de accidentes, diez fragmentos de diferentes historias que no llegan a finalizar.

Sin duda, estas obras de Italo Calvino, así como el resto de las obras literarias que se han presentado en esta sección, y muchas otras que no se han incluido, se revelan como ejemplos extraordinarios de antecedentes literarios de los modelos de creación hipertextual.

2.3.2 Los orígenes

A finales de la década de los 80 los primeros autores de hiperficción, apoyándose en las ideas promulgadas por filósofos postestructuralistas como Barthes, Derrida o Foucault y su defensa de la obra abierta, comenzaron a experimentar con el hipertexto con la finalidad de desafiar la linealidad de la narrativa tradicional.

En aquellos años se introdujeron en el mercado una serie de programas que permitían la manipulación de los textos y la creación de enlaces de una manera intuitiva. Estos programas ya contaban con una interfaz gráfica amigable que permitía al usuario un fácil manejo de la aplicación sin la necesidad de que éste conociera los secretos de la programación. Entre estas herramientas se encuentran, por ejemplo *HyperCard, SuperCard,* o el sistema *Intermedia* (una red hipertextual utilizada por los profesores y estudiantes de la *Brown University).*

Sin embargo, los primeros textos aparecidos se centraban en exceso en la idea de obra abierta que promulga la hipertextualidad y no presentaban ninguna línea argumental definida. Se consideraba que lo más relevante era el hecho de que el texto se iba construyendo a medida que el lector lo recorría, tomando sus decisiones de navegación; el lector quedaba relegado a un segundo plano.

Como consecuencia, estos primeros hipertextos no presentaban una estructura definida, y esto planteaba algunos inconvenientes. Uno de ellos consistía en la posible desorientación del usuario, motivada por esta ausencia de estructura o por el desconocimiento de la misma. El otro problema consistía en que no se explotaban todas las posibilidades que el hipertexto ofrece, y muchas de las obras constaban únicamente de un texto principal al que se le asociaban referencias bibliográficas, gráficos o anotaciones.

La potencia del hipertexto no fue realmente aprovechada hasta comienzos de la época de los 90, cuando fueron muy numerosos los autores que se atrevieron con la literatura hipertextual y la llamada hiperficción. En 1987 Stuart Moulthrop convierte al formato digital algunas de las obras de Borges y las recopila en *"Forking Paths: An Interaction Alter Jorge Luis Borges"*. Tres años más tarde, Michael Joyce publica *"Afternoon, A story",* probablemente la novela interactiva más citada y analizada de los últimos años. En 1991 Moulthrop prueba nuevamente con "*Victory Garden*", una de las hiperficciones más interesantes publicadas. Por esa misma época Deena Larsen saca a la luz dos obras de digna mención en este pequeño repaso: "*Marble Springs*" y "*Nine Vicious Little Hypertexts*". Se vive por entonces el momento de mayor entusiasmo con el nuevo género.

"*Afternoon, a Story*" es una de las obras más célebres, analizadas y discutidas de la hiperficción, así como un referente para la misma. Muchos son los artículos y reseñas escritos sobre esta novela, así como numerosas las tesis basadas en ella y en lo que supuso para el desarrollo de la narrativa hipertextual. Por todo esto, la obra merece una atención especial en este capítulo.

Esta novela, ofrecida originalmente como ejemplo para el programa de creación de hipertextos *Storyspace*, narra la historia de un hombre recientemente divorciado que ha

visto un accidente de coche y piensa que su hijo pudiera estar herido o incluso muerto a causa del choque. Está compuesta por 500 nodos diferentes, y cada una de las palabras contenidas en cada uno de los nodos es un enlace. Muchas de estas palabras, en particular artículos, preposiciones, etc. llevan a una misma página cuyo texto es *"I want to say I may have seen my son die this morning"*. El lector navega a través de los distintos nodos sin saber nunca si ha llegado al final.

2.3.3 La actualidad

Ya se ha mencionado con anterioridad que en literatura se han hecho intentos de romper la linealidad tradicional y de aumentar el grado de interactividad de los lectores, pero el hipertexto permite incorporar los saltos de página y las digresiones pueden incorporarse de modo real a la estructura de las obras. Esto es posible gracias a que el hipertexto obliga a elegir trayectos de lectura y a establecer relaciones constantemente, de modo que aunque las secuencias aisladas sean lineales, la lectura en sí no lo es, ya que los caminos no están determinados de antemano.

Los ejemplos de ficción hipertextual que podemos encontrar actualmente en la red son de dos tipos, que a propuesta de Michael Joyce en [14] muchos autores distinguen como "hiperficción explorativa" e "hiperficción constructiva". La principal diferencia entre ambas consiste en que la hiperficción explorativa tiene un solo autor y la constructiva tiene varios, por lo que se requiere una colaboración por parte de cada lector que diluye los límites entre las figuras de autor y lector.

Se va a tratar cada uno de los dos tipos de ficción por separado ya que, a pesar de que ambas derivan del texto lineal clásico, tienen antecedentes inmediatos diferentes. Por otro lado, es importante volver a mencionar que ambos tipos de ficción pueden incorporar elementos multimedia como sonidos, imágenes, etc. a pesar de que nos centraremos exclusivamente en el texto para analizarlos.

2.3.3.1 Hiperficción constructiva

La hiperficción constructiva o autoría en colaboración funciona como las IRC o *"Internet Relay Chat"*, que son charlas por escrito a través de la red en las que varias personas pueden comunicarse a la vez para escribir entre todos una historia. A diferencia de los chats actuales en los que la motivación es mantener una conversación a título personal, en las IRC también se podían mantener conversaciones en grupos o individuales, pero cada persona adoptaba de alguna manera un determinado papel en la historia que se pretendía narrar, escribiendo de esta manera un relato de manera conjunta.

El antecedente inmediato de este tipo de experiencia son los juegos de rol, que desde los años 70 ofrecen un tipo de entretenimiento creativo con postulados muy parecidos a la

liberación que propugnaban los entusiastas del hipertexto. Extraído del libro de reglas del juego de rol *Vampire* se lee: *"Long ago, before movies, TV, radio and books, people used to tell stories. Tales on the hunt, legends about the gods and the great spirits, or gossip about the affaire of others all drew rapt attention. They would tell these stories aloud, as part of an oral tradition of storytelling, but this tradition has been mostly lost. Other forms of storytelling have taken its place.*

We no longer tell stories, we listen to them, we sit passively and wait to be picked up and carried to the world they describe, to the unique perception of reality they embrace. We have become slaves of our TVs, and passively permit others to describe our lives, our culture and our personal reality through the stories that are constantly being told.

However, there is another way. Today the ancient art of storytelling has been rediscovered. A new movement is slowly growing. People are bringing stories home, making the ancient myths and legends a more substantial part of their lives. Storytelling on a personal level, rather than on the big screen or on TV, has become increasingly a part of our culture. That is what this game is all about, not stories that will be told to you, but stories that you will tell yourself." (Rein-Hagen, 1991:20)

En este tipo de juegos, un narrador o *storyteller* prepara el esquema de una historia y ejerce de árbitro regulando las acciones de los jugadores, que actúan como personajes de la historia. Los jugadores hablan y van solucionando los conflictos que les plantea el narrador como si fuesen actores en una película que tienen que ir inventando el guión a medida que se desarrolla la acción. Tienen la posibilidad de decidir sobre las acciones de sus personajes, que suelen resolverse con tiradas de dados u otro sistema similar. De este modo, a pesar de que se debe seguir un esquema básico de posibles acciones y encuentros preparados por el narrador, que controla hasta cierto punto el desarrollo de la historia, ésta se cuenta entre todos.

Si se prepara un buen esquema, las reacciones de los jugadores serán más ricas y la historia final resultante más entretenida para todos. Los libros de reglas proporcionan criterios básicos para resolver acciones que simulan las del mundo real (por ejemplo, explican qué puntuación debe sacar con el dado un jugador que quiera saltar una grieta de dos metros), evitando arbitrariedades e introduciendo el azar en el juego.

Esta intención de contar una historia en común es la misma que guía la hiperficción constructiva. No es necesario que los participantes sean personajes de la historia que se está contando (como en los juegos de rol), pero siempre hay alguien que de alguna manera ejerce un poco más de control sobre la historia, por ejemplo, integrando los diferentes textos de los participantes, etc. así como unas reglas mínimas que dirigen el intercambio y la interacción.

La hiperficción constructiva es un ejemplo de autoría compartida cuya intención es más lúdica que estética, a pesar de que al realizarse por escrito permite superar la improvisación y poca elaboración de las intervenciones personales en una partida de rol. Como contrapartida, se pierde la interacción inmediata y visual de los juegos de rol, que dan mucha importancia a la recuperación de la narrativa oral, aunque es posible que con el tiempo se llegue a experimentar con sistemas de video.

Esto no significa que la autoría unipersonal sea imprescindible para que los textos tengan un valor estético, ya que hay ejemplos en literatura que desmienten esta afirmación, pero sí que la heterogeneidad de las aportaciones dificulta el poder mantener la continuidad y calidad necesarias para que una obra sea considerada "literaria". Esto no constituye una valoración negativa de este tipo de obras en colaboración ya que su intención no es pasar a la historia de la literatura, sino estimular la creatividad y dar la oportunidad a los lectores de convertirse en autores.

Como ejemplo pionero de hiperficción constructiva, se suele citar la creación *"Hypertext Hotel"*, un espacio electrónico de creación colectiva se llevó a cabo durante años en el laboratorio de escritura creativa de la *Brown University* y que fue iniciativa de uno de los autores más conocidos dentro de este ámbito, Robert Coover.

En lo referente a páginas escritas en español, se puede comprobar como este tipo de ficción está proliferando actualmente de manera extraordinaria. En la actualidad, se pueden encontrar en la web las denominadas "wikinovelas", un formato de literatura participativa en el que se da la creación colectiva con el fin desarrollar un solo texto con todas las tramas, personajes y rutas que elijan sus autores. Vamos a ver algunos ejemplos.

Hace algunos años se puso en práctica en la web un proyecto bajo el nombre de *"La novela del 2000"*, que consistía en crear una novela colectiva con la colaboración de los usuarios de la página. Esta novela llevó por título *"La rebelión de los delfines"*, y comenzó a escribirse el 31 de mayo y se terminó el 15 de noviembre de 2000. Las normas del juego establecían de antemano que determinados capítulos (el primero, el último y todos los múltiplos de cinco) fueran elaborados por escritores profesionales de éxito, entre los que se encontraban Francisco Umbral, Espido Freire, Carmen Rigalt, José María Merino, Eduardo Mendicutti y Javier Tomeo. El resto de los capítulos serían escritos por aficionados. Durante veinticinco semanas, todos los miércoles aparecía un nuevo capítulo en la web. Este capítulo era escogido por el crítico literario Santos Sanz Villanueva de entre todos los recibidos durante la semana. La novela resultante, después de aparecer íntegramente en la web, fue publicada en forma de libro.

El argumento que resultó de este experimento en colaboración resultó ser sumamente disparatado. Se cuenta la delirante historia de un periodista llamado Walter que ha de investigar el asesinato de la esposa de su mejor amigo, supuestamente a manos de un delfín llamado Leopardo. Cada uno de los colaboradores parece querer llevar la historia a una situación desquiciada y absurda dejando en un aprieto a su sucesor. Como ocurre en casi todas estas creaciones colectivas, la intención perseguida era fundamentalmente lúdica y experimental. El crítico que ejercía el papel de árbitro dice en el prólogo que el mayor atractivo de esa creación consistía en que era un "muestrario de modos de escritura y de modelos narrativos actuales".

"Vidas prodigiosas" es una novela interactiva que estuvo alojada en la Wikinovela hasta su conclusión. El proyecto ya está cerrado, pero mientras estuvo abierto, esta obra permitía agregar nuevas entradas y hacía que fueran los propios usuarios los que aumentaran, desarrollaran y profundizaran más en cada una de las historias básicas que planteaban. De esta forma, se puede hablar de dos narradores diferentes, el propio narrador, que inició

la historia y sentó unas bases, a partir de las cuales el "segundo narrador", es decir, los usuarios, continuaron esa historia y fueron dándole diferentes enfoques, matices y nuevas tramas y personajes.

El resultado es una obra muy extensa y variable, que por momentos se hace difícil de leer debido a la gran cantidad de enlaces existentes a diferentes *historias dentro de la propia historia*. Esto es en parte debido a que en su día, esta obra fue de contenido totalmente abierto, es decir, no sólo se podían añadir nuevas entradas y enlazar páginas a la novela, sino que incluso se podían editar entradas anteriores.

Algunos otros ejemplos son las secciones de novela y guión interactivos de la publicación *"Caminos de Pakistán"* (http://caminosdepakistan.com); o *"Textos Caducos"* (http://es.geocities.com/textoscaducos/_notes/), un sitio web que da cabida a la creación de textos colectivos.

En último lugar, es interesante mencionar que, a pesar de que su autoría no es necesariamente en colaboración, también pertenecen a este tipo de hiperficción las experiencias literarias que se están llevando a cabo con los *weblogs*. En la actualidad, el *blog* se ha convertido en un espacio de encuentro digital entre el escritor y el lector, espacio donde el autor puede escribir su obra mientras que los comentarios de los lectores pueden orientar en un momento dado la reedición del texto.

El empleo de un *blog* como herramienta literaria facilita al autor el poder incluir enlaces en cada post que establezcan salidas naturales hacia otras partes de la historia. Del mismo modo, los *blogs* ofrecen la posibilidad de usar los sistemas de categorías para confeccionar mapas que "fotografíen" la narración: un sistema que permita la asignación de categorías múltiples a cada post puede referenciar al lector toda la información complementaria acerca de un personaje, una trama, un escenario, o del propio texto que se quiera añadir.

La blogosfera constituye ya un nuevo universo de muestras concretas de narrativa hipertextual. Este tipo de literatura es capaz de llegar de forma masiva a los lectores por diversas razones: porque sus límites son infinitos, porque los lectores están ávidos de una nueva forma de ficción no lineal, y sobre todo, porque la interacción y el *feed-back* se convierte en una fuente inagotable de aprendizaje, tanto para el lector como para el autor.

Las páginas que contienen diferentes experiencias de creación literaria online se multiplican sin pausa. Los *blogs* pueden aparecer ante los ojos del lector de diferentes formas: como páginas personales que cuentan ficciones como si fueran historias reales, páginas donde los poetas escriben colectivamente y aprovechan todos los recursos de la no linealidad, espacios donde los aficionados a los relatos breves comparten sus cuentos, etc...

Por otro lado, si el autor quiere aprovecharse de la realidad, la ficción online le permite insertar al personaje en el tiempo del lector de un modo creíble. El hecho de poder recibir comentarios sobre el texto es otra de las ventajas que aportan los *blogs*, sin olvidar la inmediatez con la que el texto llega al lector, así como el hecho de que no existen límites de extensión ni censura.

Algunos buenos ejemplos de experiencias literarias con blogs son *"Diario de un ama de casa"* o *"42 y 95"*, por eso merecen ser analizados un poco más en profundidad.

"Diario de un ama de casa" recibió numerosísimas visitas en pocos meses, y fue catalogada por su autor, el escritor y periodista Hernán Casciari, como un *"experimento de ficción"*. Utilizando el recurso de la bitácora, Casciari contó una historia costumbrista desde el punto de vista de un ama de casa argentina de clase media, generando la sensación de que lo que se estaba leyendo era realmente el diario íntimo de Mirta Berlotti. *"Es increíble las manera que tiene la vida de avisarte que ya no sos tan tan pobre como antes... Del cielo te cae sirvienta. ¡Mirálo vos al Kirchner, que atento!"* escribe Mirta el día en que su marido la convence de que esa señora que "ayuda" en la casa es una empleada doméstica. Y el comentario es celebrado por los lectores, que también le advierten los riesgos del despilfarro. Aunque Mirta tuvo un nieto y dejó de escribir para cuidarlo, su historia quedó registrada en http://mujergorda.bitacoras.com

"42 y 95" ,título que remite a la distancia de un maratón convencional, es también la tercera novela del autor español Miguel del Fresno, y su primera experiencia de creación literaria online, en tiempo real. Su protagonista es un corredor que describe el aprendizaje físico y mental que debe realizar para completar una carrera de fondo. *"Tenía bastante material bruto pero no encontraba el método ni el tono ni el estilo. En el formato blog encontré la solución"* explica del Fresno. *"Y no es descabellado decir que mi novela ha llegado más lejos que la mayoría de las que se publican en España. Escribí cada día, como en diario personal, mil palabras por cada kilómetro/capítulo"*. Mil fueron también los lectores que acompañaron al autor cada día durante los siete meses que duró la experiencia. La novela terminada se puede leer en http://maraton.blogspot.com

2.3.3.2 Hiperficción explorativa

La hiperficción explorativa tiene como principal característica el hecho de tener un único autor. Esto ya ocurría como norma general en la ficción lineal, sin embargo, existe algo que el hipertexto parece poder ofrecer al lector y las historias lineales no. Se trata de la oportunidad de influir en el desarrollo de los hechos que conforman la historia, o al menos, en el orden en que esta "información" le será presentada al lector. De este modo se permite al lector tomar decisiones sobre sus trayectos de lectura, eligiendo qué asociaciones establecer en cada momento.

Esto exige una actividad constante que de alguna manera aproxima los papeles de autor y lector sin llegar a confundirlos, ya que, a pesar de que los enlaces se pueden elegir libremente, todos han sido previamente pensados y escritos por un autor. De esta manera, el autor no pierde totalmente el control de la narración, como sucedía en la hiperficción constructiva. Aquí el lector no escribe, sino que decide sobre lo que ya está escrito. La clave del éxito o fracaso de este tipo de narrativa parece residir en conseguir captar y conservar el interés del lector con las posibilidades de elección que se le presenten, compensando con esto la naturaleza enmarañada y no argumental inherente al concepto de hipertexto.

Este tipo de lectura en la que se obliga a la toma de decisiones ha sido comparada muchas veces con los libros de *"Elige tu propia aventura"*, también pensados para obligar a sus lectores a tomar decisiones que determinarán el desarrollo de la aventura. Pero la hiperficción es algo más que esto, ya que la linealidad (inicio – continuación- desenlace), como idea básica y tradicional de estructura, cambia desde el principio para dar paso a las nuevas formas.

Sin embargo, la realidad está aún lejos de la teoría. Al igual que ocurría en los libros de "*Elige tu propia aventura*", los teóricos que se han ocupado de la hiperficción explorativa encuentran una serie de fallos en las obras actuales, entre los que se encuentra lo limitado de las decisiones que el lector puede tomar y que le provocan la necesidad de explorar todas las posibilidades que se le plantean para no tener la sensación de "estar perdiéndose algo".

Casi todos los críticos coinciden en que la mayoría de los intentos en esta dirección no son en el fondo distintos de la ficción lineal, de la que se diferencian sólo por ofrecer la posibilidad de hacer clic en el nombre de un personaje o cualquier otra palabra destacada sobre la que se ofrece más información al modo de una nota a pie de página o al final. Estos enlaces suelen ser bastante rudimentarios y distraer la atención de la historia principal. Esto se debe, en multitud de ocasiones, a que la historia principal está narrada de forma lineal.

Otra de las críticas más habituales que se refieren a este tipo de ficción tiene que ver con la naturaleza no lineal de la estructura del hipertexto. Se trata de la sensación de pérdida que los lectores sienten ante la multiplicidad de caminos posibles. Esto provoca que el lector tenga la sensación de no estar siguiendo "el camino correcto", a pesar de que uno de los postulados más importantes de la hiperficción es, precisamente, que no existe un único camino correcto.

Esta es la razón por la que se hace necesario dotar a los hipertextos de una estructura bien diseñada y organizada que responda a una estructura conceptual. Son necesarias herramientas de ayuda y de navegación, índices, buscadores, etc. que permitan que los lectores puedan elegir su propia ruta de entre todas las posibles sin experimentar esa sensación de pérdida. Una buena forma de conseguir esto consiste en crear la estructura de manera que la organización de los nodos y las conexiones asociativas entre ellos estén determinadas por la propia estructura semántica de las palabras que actúan como nexos o enlaces, o de una forma más general, de la estructura semántica de la propia obra.

A este respecto Jurgen Fauth realiza un análisis interesante en su artículo *"Poles in your FACE: The Promises and Pitfalls of Hyperfiction"* (ver [8]). Para Jurgen Fauth, el entusiasmo por el propio medio hace que se preste muy poca atención a la calidad del contenido de las historias. Para lograr hiperficción de calidad es importante tener presente que lo que coordina todo son los enlaces, que Fauth entiende acertadamente como la característica esencial del hipertexto, ya que se cree que la sensación de desorientación ocurre porque las opciones no son capaces de implicar al lector. No se trata de que haya

muchos, sino de que los que haya estén hechos a conciencia, de tal forma que lo que se encuentre tras cada uno de ellos no defraude las expectativas del lector.

Es necesario que los autores se den cuenta de las implicaciones del significado de los enlaces y que prevean estas implicaciones para que los lectores no se encuentren con enlaces irrelevantes. También se deben evitar los enlaces mal relacionados, de forma que no se de el caso de que conduzcan a textos absolutamente inesperados en el trayecto de lectura personal. No basta con ofrecer elecciones por el mero placer de elegir ya que la novedad de "hacer clic" no dura mucho tiempo.

Las elecciones simples o pragmáticas fueron ya anteriormente rechazadas por Michael Joyce, quien en *Afternoon: a story* [30], propone un nuevo tipo de participación del lector que llama *"words that yield"* y que busca un desarrollo del propio lenguaje de la historia. De esta forma, en la frase de *Afternoon "I want to say I may have seen my son die this morning"*, el lector seguirá un hilo argumental al seleccionar la palabra *son* diferente al que seguiría si seleccionara la palabra *die*.

El intento de Michael Joyce se parece mucho más a los experimentos de las vanguardias literarias, y en obras como "*Finnegans Wake*" [13] o *"At Swim-Two-Birds*" [20] se pueden encontrar apuntes sobre por dónde pueden ir las posibilidades más atrayentes de la hiperficción.

A este respecto, es interesante tener en cuenta las recomendaciones de Sarah Auerbach en su artículo *"Hypertext Fiction: A Literary Theory"* (ver [2]). Ella plantea que el éxito de este tipo de ficción pasa por las siguientes directrices: el texto debe ser breve, *"ya que las ramas en expansión son un gran problema para autores hipertextuales, es importante centrarse en un asunto pequeño y expandirlo hacia la complejidad infinita y no empezar por el mundo y luchar luego por expresarlo en nexos cada vez más raquíticos"*. Los personajes deben ser lo más importante y no se debe tener prejuicios acerca de lo que el hipertexto puede o no puede hacer, ya que, al igual que la ficción lineal, puede hacer lo que quiera mientras que lo haga bien.

En ese mismo artículo se afirma que gran parte de la hiperficción actual tiene como sello característico el hecho de que sus autores intentan conservar elementos propios de la ficción lineal como son el argumento y los personajes, dando la impresión de que el hipertexto, en general, todavía depende del género del que nació. Sin embargo, esto puede y debe cambiar. La hiperficción tendría que ser lo que sostiene ser, es decir, una mejora de la ficción lineal, y con este fin debería conservar lo mejor que tiene este tipo de ficción y deshacerse de lo peor de ella.

En este punto de la reflexión cabe preguntarse qué es lo mejor de la ficción lineal, teniendo en cuenta que esto depende del género en concreto del que se esté hablando. En lo que a la narrativa lineal se refiere, lo mejor es su capacidad de capturar la realidad en conjuntos manejables; hace posible experimentar la realidad y el punto de vista de otro, vivir la vida de otro, experimentar las emociones de otro. La hiperficción debe preservar estas características.

En el extremo opuesto, y tomando como apoyo las ideas desarrolladas en [2], se puede aventurar que aquello que la hiperficción puede permitirse dejar atrás es el argumento como estructura. La ficción lineal mantiene el interés de los lectores en parte gracias al suspense, como menciona Landow en [16], por la curiosidad de saber "qué pasa al final". La ficción hipertextual puede abandonar las ideas de principio, medio y final. La ausencia de linealidad implica que el argumento no es algo fijo, sino que se va construyendo en cada lectura, y que las propias relaciones entre los distintos fragmentos de texto son determinadas por el lector. Es evidente que no se puede escribir hiperficción con el pensamiento de una única línea causal que deba ser "encontrada", sino que los distintos nodos han de permitir cualquier combinación dirigida por una búsqueda de sentido personal.

Si la hiperficción se queda estancada en el uso de la idea tradicional de argumento, probablemente desembocará en historias del tipo *"Elige tu propia aventura"*, que a pesar de que puede ser el ejemplo de hiperficción más efectivo, al menos en los inicios, son historias que pueden resultar divertidas pero a la larga escasamente satisfactorias.

En este momento es complicado aventurar algo que pueda ocupar el lugar del argumento, pero es necesario que los autores de hipertexto encuentren algo que sea tan convincente o más para dar forma a retazos de vida y que se adapte a la forma no lineal, sin olvidar que lo predecible aburre y que el lector no está dispuesto a aceptar lo inverosímil. Es probable que los personajes tengan que ser doblemente fuertes para soportar una historia sin argumento o con un argumento no lineal.

A continuación se analizan algunas obras, ejemplos de la actual hiperficción explorativa, que se pueden encontrar colgadas en la red:

"El reino de los espejos torcidos" es una historia de hiperficción explorativa alojada en http://www.unav.es/digilab/proyectosenl/0001/final/espejos/index.htm. Su interfaz de comienzo dirige al lector a otra página a modo de menú a partir de la cual es posible entrar en la historia. El lector pude elegir entre el punto de vista de dos personajes diferentes, avanzar o volver al inicio para escoger opciones que ha dejado atrás para poder ver la historia de otro modo.

Es un relato no lineal porque se desarrolla en un esquema de dos líneas en paralelo. En cuanto a los enlaces, existen tres en cada página: inicio, siempre para poder volver, y según la página, avance y una de las dos visiones de la historia (o las dos): "Olga" y "Aglo".

Es un relato en el que existe un único autor, arquitecto y director de su propia obra, donde el usuario que entra asume el punto de vista que el escritor propone (los dos puntos de vista de los personajes principales) sin darle demasiada libertad de imaginar. Por lo tanto, esta obra constituye un universo clausurado con opciones limitadas.

"El aprendiz del detective" presenta una trama que se centra en resolver un caso de asesinato, identificando al usuario con el detective encargado de ello. (http://www.unav.es/digilab/proyectosenl/0001/final/aprendiz_detective/index.htm)

Esta obra posee un carácter interactivo indiscutible, puesto que el usuario es el que da comienzo y continuidad a la historia. Sin embargo, las posibilidades de elección que se le plantean al usuario no consiguen implicarle de manera suficiente en el relato. La hipertextualidad existe, pero de un modo bastante reducido, ya que aunque en el comienzo se supone que el usuario se irá encontrando con diferentes caminos dependiendo su elección, en un punto bastante cercano al inicio, esto desaparece y con ello la sensación de independencia del usuario. En este momento pasa a someterse completamente a la tiranía del diseño del autor, que ya sólo presenta un hipertexto por pantalla y la única posibilidad es seguir hacia delante. La historia se convierte en una narración sin diferencia a la puramente lineal, ya que utiliza una estructura de libro impreso, en el que para continuar la única opción es pasar las hojas.

De esta manera, el autor de la obra se presenta como un único responsable de su desarrollo, dejando al usuario en un segundo nivel, casi de mero observador, aunque se le presente como el protagonista, lo que resulta ser un engaño. En esta situación, el usuario siente una profunda decepción, ya que se le ofrece la ilusión de haber entrado en un universo abierto en el que interactuar con los personajes y poder construir una trama, para arrebatársela rápidamente.

Por todo lo dicho, *"El aprendiz del detective"* es una obra muy cerrada, que carece de opciones de navegación más allá de la principal, lo que unido a un diseño muy simple hace que no resulte un proyecto del todo interesante, debido en gran medida a la escasez de opciones que ofrece al usuario unido a su falta de trama.

Tal vez todas las objeciones que se han puesto hasta aquí a las hiperficciones analizadas sean justificables por el hecho de que nos encontramos en una época de transición y experimentación. De cualquier modo, lo que verdaderamente esta ficción aporta de novedoso desde el punto de vista creativo no es tanto la idea que subyace bajo el proyecto, sino la posibilidad, dada por la técnica, de materializar esa idea.

Para concluir, es importante recordar lo que se ha comentado al comienzo del capítulo: lo que persigue la hiperficción o narrativa hipertextual es algo más que el simple hecho de vincular documentos con herramientas informáticas. Son necesarias *organizaciones hipertextuales*, elementos multimedia, formas creativas de narración y herramientas que hagan más sencilla la confluencia entre tecnología y literatura y que permitan la experimentación.

En otras palabras, es necesario llevar la teoría a la práctica, y convertir la narrativa hipertextual en verdadera innovación, esto es, en un elemento presente en el día a día de los lectores. De esta forma, se estarían propiciando las condiciones básicas necesarias para que este nuevo género vaya tomando forma, haciendo finalmente posible lo que Rodríguez de las Heras denomina en [25] *"un libro sin páginas, un libro blando, poliédrico, navegable, con emociones nuevas de lectura…"*.

2.3.4 El futuro

La situación descrita en el apartado anterior relativa a la actualidad de la hiperficción dibuja una nueva forma de arte que se encuentra aún en sus inicios. Pero esta situación podría continuar así aún durante un largo periodo de tiempo, ya que se trata de una materia en relación con el mundo de las humanidades, en la que un avance planificado y controlado resulta complicado de llevar a cabo.

Las posturas de los diferentes autores y teóricos ante este escenario que se plantea en la actualidad pueden agruparse fundamentalmente en dos posturas de signo contrario. La primera de ellas consiste en esperar hasta que las bases sobre las que se sostiene la narrativa hipertextual evolucionen hasta alcanzar un estado de madurez suficiente como para producir obras de elevada calidad. La segunda apuesta por acercar este tipo de literatura a la sociedad de manera que su consumo se generalice, lo que permitiría obtener realimentación de la misma, acelerando, por lo tanto, su proceso de evolución de manera significativa.

A lo largo de este documento se defiende la segunda de las opciones presentadas. Howard S. Becker y su artículo *"Ficción hipertextual, una nueva forma de arte"* [3] son una referencia muy sólida a este respecto. En este artículo se afirma que los trabajos artísticos involucran la cooperación de distintas actividades para alcanzar el resultado óptimo. Esto incluye a las personas que crean las obras, las editan o las distribuyen. Incluye también a la crítica y por supuesto, también al público. De esta forma, una nueva forma de arte no es solamente el hallazgo de nuevas expresiones por parte del artista, sino la extensión de su uso y la dinamización de todas aquellas actividades complementarias que la hacen posible.

Por este motivo es importante que esta nueva forma de literatura pueda empezar a llegar al gran público, a pesar de que los intentos actuales de aprovechar las ventajas de la tecnología para enriquecer la literatura no estén aún a la altura de los logros alcanzados por la literatura tradicional. Lo más importante en este momento en el que se encuentra la hiperficción no es crear obras maestras, sino popularizar los productos de la narrativa hipertextual.

Sin embargo, a pesar de que estas primeras obras estén destinadas a familiarizar al público con las dimensiones que los nuevos medios pueden aportar a la narrativa y que la hagan dar un paso importante en su evolución, es importante llevar a cabo este proceso de una forma cuidadosa. Es decir, estos aportes nunca deberían poner en peligro los logros alcanzados por la literatura tradicional desde sus orígenes, ya que esto supondría un retroceso (y no un paso adelante) para la literatura.

José Jesús García Rueda y Carolina Franco Espinosa hacen una reflexión interesante sobre este tema en su artículo *"Narrativa hipermediática"* ya referenciado con anterioridad [9]. En su opinión la sociedad actual presenta "*claros síntomas de mecanización de lo humano frente a una mucho más deseable situación de humanización de lo maquinal"*, lo que supone una amenaza real para las obras literarias. En una sociedad como ésta, en la que las personas se dejan arrastrar fácilmente por el dinamismo, la interacción y la capacidad audiovisual de los nuevos medios, no es difícil que se pierda la verdadera esencia de la literatura, esto es, el elemento humano. Esto que se ha alcanzado durante siglos mediante

el uso de la página impresa, no puede perderse como consecuencia de la implantación de las Tecnologías de la Información y las Comunicaciones.

La siguiente figura, incluida en el artículo, muestra la manera en la entienden los autores que podría producirse la coordinación necesaria entre los elementos que aporta la literatura tradicional y las novedades introducidas por la hiperficción para dar lugar a nueva forma de literatura.

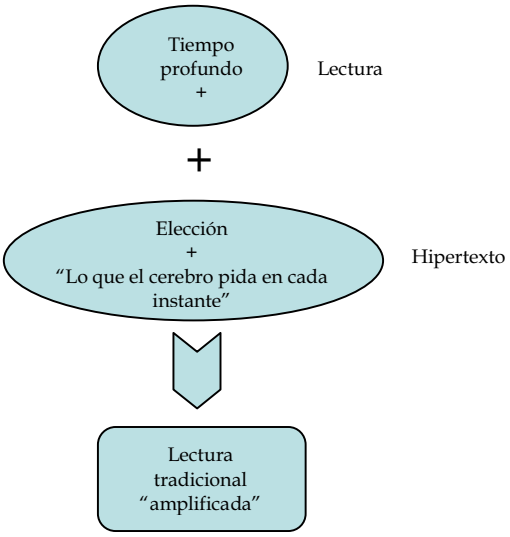


Figura 2.1 Aportaciones del hipertexto.

El lector se aproxima a una obra literaria por motivos que pueden ser muy diversos, basándose en intereses que pueden ir desde la búsqueda de entretenimiento hasta el análisis en profundidad de una obra. Pero de entre todos los elementos que una obra literaria puede aportar a su lector, en el artículo se destaca lo que Birkerts en [4] denomina "tiempo profundo" y que se define como *"un estado de comunión íntima con el texto. Un fundirse con lo que se lee, penetrando en profundidad en el universo que las palabras están abriéndonos, y a la vez dejar que dicho universo penetre en nosotros. Hacer, en definitiva, que al menos mientras leemos, el tiempo pase según el ritmo que marca nuestro interior y la obra (uno dentro del otro y viceversa). Marcar nosotros el ritmo de la secuencia. Esto vendrá de la mano, las más de las veces, de una fuerte introspección: la lectura, el libro, nos introduce en nuestro interior y extrae de él algo que ya estaba allí, pero de cuya presencia no éramos plenamente conscientes"*.

Por lo tanto, la existencia de este tiempo profundo implica que la lectura en un buen lector nunca es pasiva, siempre es profunda, sin que la obra sea necesariamente interactiva. Y la narrativa hipertextual debe ser capaz de favorecer esa profundidad. Se debe hacer un esfuerzo por trasladar este logro de la narrativa a los nuevos medios, porque en él radica gran parte de la grandeza de la literatura.

De este modo, cualquier elemento de decisión, de interacción, de visualización y de inmersión que se introduzca en la narración debe ir siempre encaminado a potenciar el tiempo profundo. Las inmensas posibilidades que aporta la hipermedia pueden y deben usarse para aumentar la fusión entre relato y lector, y nunca al contrario.

Sin embargo, esto es algo que el hipertexto no fomenta demasiado debido, entre otras cosas, a que los cambios constantes de contexto pueden llegar a romper el ritmo interno de lectura, lo que provoca una falta de permanencia de los contenidos en el lector. En ocasiones puede incluso ocurrir que se favorezca una profundidad engañosa, ya que el lector se cree implicado en la historia porque participa de ella, pero realmente no está interiormente implicado en su desarrollo.

Por otro lado, existe la denominada "paradoja de la libertad de decisión", consistente en que, aparentemente, una narración hipertextual ofrece al lector muchas más posibilidades de elección que la narrativa tradicional. Sin embargo, lo que realmente sucede en muchas ocasiones es que las infinitas opciones implícitas que proporciona la libertad de hacer tuyo el universo de una obra, interiorizando todas las emociones que ésta transmite, se ven reducidas con el hipertexto, que simplemente ofrece *"un conjunto discreto de opciones explícitas"*. Es necesario conseguir que las obras de la narrativa hipertextual creen un mundo en el interior del lector, al igual que hacen las obras de la narrativa tradicional, y que todos los elementos hipermedia ayuden a mostrarle al lector ese mundo que ya se ha creado en su interior.

En este sentido, García Rueda y Franco Espinosa apuestan por una aproximación progresiva al problema. Esto significa que no es necesario que ya desde los comienzos los relatos hipertextuales sean capaces de adaptarse perfectamente a los deseos del lector, sino que lo verdaderamente importante es que el lector *"vaya descubriendo la obra y su universo, como en un libro tradicional"*. Esto permite alcanzar todas las ventajas de la narrativa hipertextual mediante la introducción de pequeñas modificaciones al libro tradicional. Se debe tener en cuenta que estas modificaciones deben ser tan pequeñas como sea necesario para garantizar que con su introducción no se pierde ninguno de los logros alcanzados ya con la narrativa tradicional.

Este proceso de modificación descrito debe comenzar con un acercamiento al gran público, y en los comienzos, el hecho de que la literatura hipertextual no se aleje demasiado de los preceptos de la literatura tradicional puede resultar positivo, ya que esto facilitará el acceso a esta nueva forma de narrar a un número mayor de lectores. Por este motivo no es malo crear productos de consumo basados en la narrativa multimedia, al contrario, es probablemente la acción correcta a realizar actualmente.

Lo que sí sería nefasto es que esta nueva forma literaria no evolucionara adecuadamente y quedara para siempre asociada a una forma de literatura menor, ya que su potencial da para mucho más que esto. El hipertexto es capaz de llevar la comunión entre lector y obra a un nivel inalcanzable por los medios tradicionales: un hipertexto permite seguir en cada momento de la historia el camino que el lector está inclinado a seguir. Esto contribuye a enriquecer, profundizar y hacer mucho más rica la interacción entre lector y obra.

Capítulo 3:
Estado actual del arte

3.1. Hiperdrama

En el capítulo anterior queda expuesto cómo el hipertexto ha revolucionado la forma de entender la novela, pero el uso del hipertexto no es algo exclusivo de este género. En los últimos años ha aparecido lo que se ha dado en denominar hiperdrama, o lo que es lo mismo, el uso del hipertexto en obras escritas con el fin de ser representadas en un teatro.

En este aspecto es de especial relevancia la obra de Charles Deemer, quien en [27] explica cómo él mismo se encuentra por primera vez en su vida ante la situación de verse bloqueado ante la pantalla de su ordenador. Sin embargo, éste bloqueo no se debía a la trama de la obra, sino a la manera en la que debería numerar las páginas. Él había imaginado una obra en la que varias escenas tenían lugar a un mismo tiempo, y la página escrita no le permitía reflejar eso sobre el papel. Más tarde comprendió que el uso del hipertexto podía solucionar su situación y desde entonces, lo emplea en sus obras dramáticas.

Como él mismo explica, el resultado de aplicar el hipertexto al género dramático consiste en una obra en la que muchas escenas ocurren a un mismo tiempo. En este caso, la audiencia, al igual que el lector de hipertexto, debe decidir "qué ocurre después". Por ejemplo, el siguiente extracto de su obra "*Chateau de Mort*" [42] tiene dos momentos de decisión para los espectadores en menos de un minuto. Dicho en otras palabras, la historia se ramifica dos veces en menos de un minuto.

(Jack, Polo, Heather and Medallion are in an upstairs bedroom.)

HEATHER: Want to go downstairs for a drink?

MEDALION: Why not?

HEATHER: Jack? I'm really sorry about your dad.

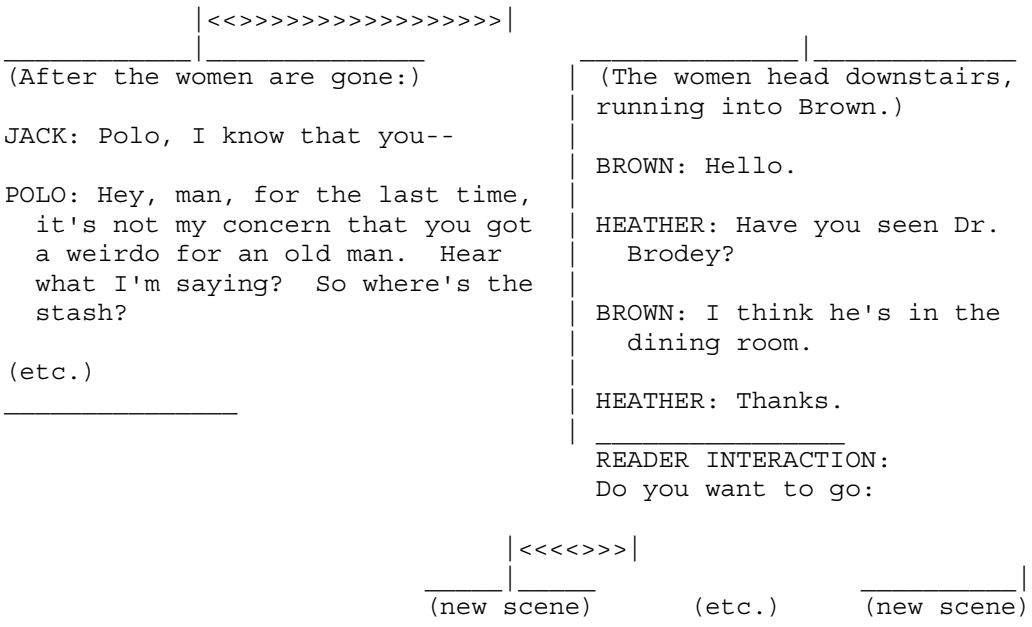
JACK: Thanks, hon.

(The women leave to go downstairs.)

READER INTERACTION:

Do you want to follow the women
or stay here with the men?

_____
Make your choice below:



En el teatro tradicional, el espectador que se sienta a ver una obra comprende por sí mismo durante el transcurso de la misma cuáles son los personajes principales y cuáles los secundarios. Aprende, del mismo modo, cuál es la historia principal y cuáles son aquellas pequeñas historias secundarias que generalmente tienen alguna relación con la principal.

La audiencia es capaz de llegar a este conocimiento gracias a que el autor de la obra ha realizado durante la escritura de la misma una serie de elecciones personales que condicionan la información que el espectador recibe dando mayor peso a la acción principal o al personaje o personajes protagonistas. Lo que se ve en el escenario del teatro tradicional es, según propias palabras de Deemer, *"una visión artística de una historia que depende de elecciones personales (qué enfatizar aquí o que subestimar allí), una modelación personal del material que llega al espectador mediante la acción lineal que se representa en escena y de la que toma acción de una forma pasiva. Yo llamaría a esta comunicación personalizada la tradicional visión única del autor"*.

El hiperdrama es capaz de desafiar esta visión única. Este nuevo tipo de teatro construido a partir de hipertextos, divide la acción principal en diferentes ramas. Las butacas estáticas son arrancadas de sus sitios para dar a la audiencia movilidad, para ofrecerles la oportunidad de seguir diferentes caminos de la historia que se está representando.

Como resultado de la aplicación del hipertexto a este género, al igual que ocurría con la novela, se produce la pérdida de muchos de sus elementos característicos. Se pierden, por lo tanto, las nociones de personajes protagonistas o acción principal, porque todos los actores permanecen en escena desde el principio hasta el final, y todos ellos tienen su propia historia, completa y desarrollada durante el transcurso de la obra.

Sin embargo, a pesar de esta aparente ruptura con el teatro tradicional, el hiperdrama no entra en competición con él, sino que se considera que se ha encontrado un nuevo modelo en el que el modelo antiguo tiene cabida como un caso especial de éste. Desde esta perspectiva, el hiperdrama consiste en una expansión enriquecedora del teatro tanto para el autor como para los espectadores, redefiniendo por lo tanto lo que significa contar una historia dramática en una actuación en vivo.

3.2. Scenejo

Scenejo es otro ejemplo más de cómo el hipertexto forma parte de la vida literaria en al actualidad. Esta herramienta se encuentra descrita al detalle en [43], aunque a grandes rasgos se puede definir como una plataforma interactiva para contar historias que soporta tanto líneas argumentales estructuradas como comportamiento emergente. En la plataforma tienen cabida varios actores artificiales conversando con un número variable de actores reales que representan los usuarios del sistema. Los actores artificiales pueden ser visualizados como personajes animados en tres dimensiones y las respuestas son presentadas mediante síntesis del lenguaje en combinación con comunicación no verbal.

El hecho de contar historias de una forma digital e interactiva tiene además el potencial de convertirse en el paradigma para los futuros medios de aprendizaje, ya que combina la narración dramática y la interacción de los usuarios, proporcionando de esta forma un nivel más alto de compromiso e inmersión en la historia.

En este sentido, *Scenejo* es capaz de soportar conversaciones entre diferentes actores artificiales en combinación con la entrada libre de texto por parte del usuario en un entorno multiusuario. Ofrece la posibilidad de dejar que los personajes virtuales entren en debate o incluso en discusión, incluyendo a los usuarios en los mismos pidiéndoles consejo, preguntándoles su opinión o instándoles a que dirijan la conversación hacia las áreas de su interés. Las áreas de contenido ideales para cubrir con estas conversaciones son las materias que contienen factores susceptibles de interpretación, como el arte o la filosofía.

Por otro lado, Scenejo proporciona también un sistema de autoría para crear y experimentar conversaciones emergentes. Los usuarios pueden crear un escenario definiendo los personajes y sus características, tales como género, voz, representación visual e incluso un diálogo base particular para cada personaje. Además, proporciona un editor para la estructura o grafo de la obra en el que se describe las posibles conversaciones o escenas de una historia. Las escenas representan los bloques con los que se construye una obra y constituyen el contexto y el entorno para una parte de la historia interactiva. En la siguiente figura se puede observar la interfaz de usuario de esta plataforma:

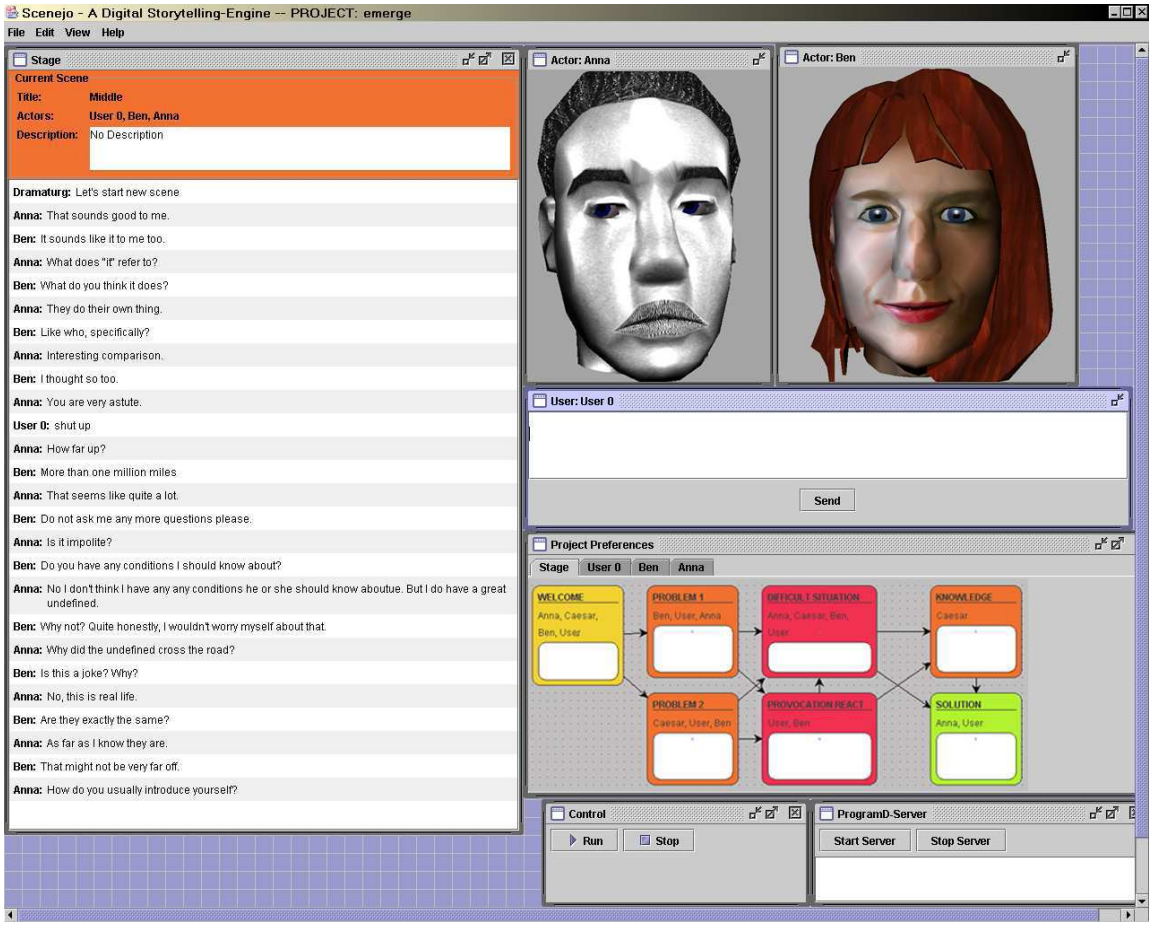


Figura 3.2 Interfaz de usuario de *Scenejo*.

Una vez que la conversación comienza, los personajes que están asignados a la primera escena empiezan a charlar en consonancia con sus patrones de texto. Dependiendo de estos patrones, que han sido previamente creados por el autor, la conversación puede derivar en un diálogo estructurado, en una argumentación caótica o en una simple charla.

El diálogo que está teniendo lugar está representado por las cabezas animadas que se pueden ver en la figura anterior, que hablan con voz sintetizada digitalmente sin ninguna necesidad de grabación previa. Adicionalmente, una representación textual de la escena proporciona una visión general del diálogo, desde el principio de la conversación hasta el punto actual.

3.3. La primera novela escrita por un ordenador

Este sea quizá, de los apartados que componen este capítulo, el que más tangencialmente tenga que ver con el hipertexto. Sin embargo, se ha considerado adecuado e interesante incluirlo en el mismo, ya que se muestra como una evidencia más del modo en que el mundo de la tecnología y las Humanidades van cada vez más de la mano.

Durante el 2008 ha visto la luz la primera novela escrita por un ordenador. La ha lanzado al mercado la editorial rusa Astrel SPb, de San Petesburgo, bajo el nombre de *Amor Verdadero y* se trata de una obra basada en la novela de León Tolstói, Anna Karenina, y ambientada en una isla desierta. El "escritor" artificial consiste en un programa informático bautizado como PC Writer 2008, al que se le puede incorporar el desarrollo de una trama, un estilo de escritura y un tiempo y lugar en los que situar la historia.

Los principales personajes de la novela imitan a los de Anna Karenina, la acción se desarrolla en una misteriosa isla, y la época escogida para la ambientación de la obra es la actual. Asimismo, la máquina ha imitado la forma de escribir de un escritor muy conocido, el japonés Haruki Murakami, autor de libros como *Crónica del pájaro que da cuerda al mundo* o *Sputnik, mi amor.*

Según declaraciones del editor de Astrel SPb, el programa fue desarrollado durante ocho meses por un grupo de creativos y filólogos. Estos últimos recopilaron historiales relacionados con cada uno de los personajes de la obra, en los que describían su apariencia, su vocabulario, su perfil psicológico y otras características. Una vez establecidas las pautas iniciales, éstas se incorporaron al programa informático que fue capaz de escribir la novela en sólo tres días.

Según los editores, este programa tiene la ventaja de que es seguro que el libro demandado estará listo a tiempo y será fiel al estilo y las características buscadas. Por otro lado, se señala que los costes de publicación de un libro artificialmente generado son más bajos que los derivados de los honorarios habituales de los escritores. Esto es un claro reflejo de cómo la tecnología no ha sido sólo capaz de revolucionar la forma de entender la literatura, sino también la forma de edición y generación de obras.

PCWriter 2008 es el primer programa informático del que se tiene noticia que sea capaz de producir una novela, aunque sea con la ayuda humana. Sin embargo, no es el primer software que se conoce creador de ficción.

A principios de 2007, la revista *Discovery Channel* hablaba de la creación de MEXICA, un programa informático que escribía sus propios relatos de ficción. Es decir, generaba historias basadas en representaciones computerizadas de emociones y tensiones entre diversos personajes. Y parece que estas historias son interesantes, ya que según una encuesta llevada a cabo en Internet para determinar la valoración de las historias "artificiales" por parte de los lectores, en comparación con la valoración de relatos escritos sólo por humanos, demostró que los lectores puntuaban más alto los cuentos de MEXICA

que los demás debido a su coherencia, su estructura, su contenidos, su suspense y su calidad en general.

3.4. HyperAuthor y su contexto

El hipertexto va más allá de una simple escritura electrónica, su no linealidad revoluciona la forma de abordar un texto, cada bloque o nodo puede involucrar una cantidad de información distinguida de un tema específico, y no sólo en un formato escrito, sino también usando imágenes, audio, video, etc. Los nodos se enlazan unos con otros formando una red con múltiples trayectos que le permiten al lector navegar en la información existente.

No obstante, existe un riesgo de perderse en un mar de información debido, de forma principal, a la variedad de caminos ofrecidos. El tamaño del hipertexto influye de forma directa en la complejidad del mismo; en hiperficciones complejas no es difícil que el lector olvide el modo de acceder a una parte del relato que sabe que existe o incluso que se dé la situación de que ignore el lugar donde se encuentra y al que ha llegado navegando.

Por todos estos motivos, es posible que la lectura o navegación por un hipertexto esté regida por el principio de indeterminación, incertidumbre, imprevisibilidad, improbabilidad, azar, caos, etc.; también puede darse la situación de que la estructura siga una estructura lineal con diferentes alternativas. En ambos casos, la lectura del hipertexto no resultaría satisfactoria. Es importante, por lo tanto, que los autores de hiperficción tengan presente durante todo el proceso de creación que la estructura de un hipertexto bien construido deber ser una estructura sistemática que obedezca a un modelo conceptual determinado. Del mismo modo resulta de gran utilidad la incorporación al mismo de herramientas de navegación o mapas conceptuales que le ayuden al lector a "orientarse" durante la lectura.

Con el fin de facilitar el correcto diseño y elaboración de hipertextos, son cada vez más numerosas en el mercado una serie de herramientas específicas. En el siguiente apartado se va a hacer un análisis de las más populares y, tomando como base este análisis de los puntos fuertes y las debilidades encontradas en cada una de las herramientas, se procederá a la concreción del modelo sobre el que se basará la herramienta a desarrollar y sus especificaciones.

3.4.1 Algunas herramientas

Como se ha establecido en el apartado anterior, las necesidades identificadas en este nuevo género son: organizaciones hipertextuales, elementos multimedia, formas creativas de narración y herramientas tecnológicas.

Las organizaciones hipertextuales aplicables al género literario pertenecen a los tipos de estructuras básicos conocidos, y con ellos es posible crear narraciones que permiten descubrir la historia, explorar secuencias alternativas, experimentar la representación de roles, conocer múltiples versiones de la historia y construir la trama. Las obras de este nuevo género deben transmitir al usuario un sentido claro de la finalidad, facilitarle una navegación transparente, comunicarle de modo efectivo la estructura (evitando la sensación de pérdida), responder a sus acciones y ayudarle a explorarla.

Con este objetivo, las herramientas deben favorecer que las obras posean las cualidades deseables enunciadas por los teóricos. Éstas son:

- Intención: la obra debe trasmitir un sentido claro de la finalidad. Por qué estoy contando esta historia (autor) y por qué estoy navegando esta historia (lector).
- Inmersión: relacionada con la experiencia del usuario. La inmersión es óptima cuando se logra captar y mantener su atención. En este punto es importante que el sistema sea transparente y que se haga un buen uso de los elementos multimedia.
- Estructura: el usuario debe ser consciente de la estructura en la que se mueve, para lo que son útiles los mapas.
- Respuesta: grado de efectividad de la interacción que permite la obra
- Guía: sistema de ayuda, que puede ser sutil o consistir en complejos sistemas de ayuda sensibles al contexto.
- Uniformidad: elección de criterios similares en la interacción
- Consistencia: acciones parecidas producen resultados similares, para no desconcertar al lector
- Simplicidad: en el planteamiento general de la navegación

En lo que respecta a las herramientas existentes, ya se ha comentado que empiezan a ser cada vez más comunes las herramientas de autor para este nuevo género, que en Internet se conoce como "escritura-no-lineal".

Con el fin de conocer sus funcionalidades y poder compararlas con los fundamentos teóricos de la hiperficción, se van a analizar las herramientas más populares que existen actualmente.

3.4.1.1 Storyspace™

Storyspace se creó como un sistema de hipertexto para Macintosh que fue diseñado expresamente para escritores. Lo crearon Jay David Bolter, un humanista que daba clases en el Instituto de Tecnología de Georgia, John B. Smith, profesor de Informática de la Universidad de Carolina del Norte y Michael Joyce, cuyo trabajo con hiperficción es tan destacable que ha sido citado en numerosas ocasiones a lo largo de este texto.

En [44] podemos encontrar una extensa descripción de este programa desde el punto de vista del usuario. En lo que sigue se exponen aquellos aspectos del mismo que se han considerado de mayor relevancia para el proyecto.

Storyspace utiliza la terminología empleada por Bolter, usando el nombre de espacios de escritura para los nodos. Un espacio de escritura podía estar formado por texto, gráficos, sonido, vídeo e incluso por otros espacios de escritura. Cada espacio de escritura se mostraba en una ventana desplegable sobre un escritorio que incluía una barra de herramientas y una serie de menús desplegables. Las opciones que estos menús ofrecían era muy limitadas, las funcionalidades de corrección de texto eran mínimas, el control de fuentes era muy sencillo y no existían ni el formato de párrafo ni el de estilo.

El espacio de escritura podía soportar sólo 32.000 caracteres. Esto era suficiente para la mayor parte de los objetivos, pero se trataba de una limitación importante, ya que además, la herramienta obligaba a dividir los archivos en nodos separados cada 25.000 bytes. El usuario podía reducir el número, o bien decidir dividir el nodo después de cada párrafo, pero ninguna de estas opciones permitía una ruptura natural entre capítulos u otras unidades, aunque la herramienta sí daba opción a guardar las preferencias del entorno de usuario.

En la actualidad, *Storyspace* sigue siendo desarrollado por *Eastgate Systems* con la supervisión de Mark Bernstein. La versión 2.0 incorpora nuevas características, como una mejor interfaz, un mejor manual y la posibilidad de hacer uso de las tecnologías actuales como arrastrar y soltar los menús contextuales.

En esta nueva versión, al texto se le puede cambiar el estilo, y la navegación mediante teclas de cursor es suficiente para que la herramienta pueda considerarse un entorno de edición competente, aunque primitivo. Cada espacio de escritura tiene un nombre, distinto del texto que lo constituye. El estilo del nombre no puede editarse, su tamaño debe ser menor de 32 caracteres y sería útil que no se repitiera, aunque no es obligatorio.

Esta herramienta organiza los espacios de escritura mediante una jerarquía, es decir, algunos espacios de escritura pueden estar en el nivel superior y algunos pueden subordinarse a otros. Al visualizar la jerarquía, se ven los nombres de los espacios de escritura; para reorganizar la jerarquía, puedes arrastrar un nombre, o hacer doble clic si lo que se pretende es descubrir el texto que contiene.

Existen cuatro formas distintas de visualizar la jerarquía de espacios de escritura:

- **Vista de esquema.** Esta vista muestra una lista vertical con los nombres de los nodos a modo de esquema. Los nodos que tienen otros nodos subordinados tienen un triángulo desplegable a su izquierda, que permite mostrarlos u ocultarlos, y existe una navegación básica mediante teclas de cursor.

- **Vista de gráfica.** Muestra los nodos en un organigrama horizontal. Como gráfica, ésta parece un árbol genealógico puesto de lado. Los nombres de los nodos del nivel superior forman una columna a la izquierda. Si un nodo tiene sub-nodos dentro de él, éstos se enumeran a su derecha, vinculados a él mediante líneas.

- **Vista de mapa.** Muestra los nodos en un mapa global en el que éstos se representan como pequeños rectángulos de tamaño fijo con sus nombres encima. Estos rectángulos representan los nodos de un solo nivel, de forma que si se selecciona uno de ellos y se pulsa la tecla de flecha hacia abajo, el rectángulo se expande hasta ocupar toda la venta y se ven los nodos subordinados a él. Si se pulsa la flecha hacia arriba se sube un nuevo nivel.

- **Vista de árbol.** Esta vista se parece a un mapa porque los nodos se muestran como rectángulos con sus nombres encima, y porque es posible expandir uno e introducirse en él, de forma que ocupe toda la ventana. La diferencia está en que se muestran todos los nodos de los niveles inferiores, con independencia de la situación actual del usuario. No hay barras de desplazamiento y los nodos adaptan su propio tamaño para caber repartidos en la ventana.

El cambio entre una visualización y otra no sucede en una misma ventana, sino que se abre una nueva ventana de visualización del tipo elegido. Es posible tener abiertas tantas ventanas de visualización como sea necesario. Cada una ofrece un menú contextual, con un listado de todos sus nodos de forma que es posible seleccionar cualquiera de ellos. En cualquiera de estas vistas se incluye también una herramienta de amplificación que permite usar el zoom sobre un área de interés particular. Una ventana aparte, la ventana *locate*, muestra un listado alfabético de todos los nodos y permite abrir una nueva vista a partir de éstos. De esta forma el usuario es capaz, casi con total certeza, de acceder y examinar de cerca todas las partes del documento de la forma que considere más oportuna y luego reorganizarlas. Es importante comentar que, al reorganizar la jerarquía en una ventana de visualización, el contenido de las demás ventanas de visualización cambia para reflejar la nueva organización de una forma automática.

El aspecto más deficiente de la organización jerárquica expuesta es la denominación de los nodos o espacios de escritura. El hecho de que los nombres estén limitados a 32 caracteres dificulta gravemente la posibilidad de designar un nodo de manera que el nombre resulte lo suficientemente informativo. De esta forma, con independencia del tamaño del documento, no hay manera de identificar los recortes solamente por su nombre.

El documento posee un segundo nivel de organización, dado por los enlaces hipertextuales. Un enlace puede emanar de un nodo completo, de un segmento de texto en el interior del nodo o de una imagen. El enlace puede conducir a otro nodo entero o a un segmento dentro de él. Los enlaces se crean de una manera muy sencilla: si los dos nodos están visibles y el enlace conduce a un nodo completo, basta seleccionar el origen, elegir "crear enlace" y hacer clic en el destino. En caso contrario, se crea el enlace hasta una

paleta flotante denominada *túnel*, y a continuación se completa el enlace desde ésta al destino.

La navegación, como se ha comentado, también es muy sencilla: en esencia, consiste en hacer clic en el origen de un enlace para abrir el texto de destino. Cuando se accede a un determinado nodo mediante un enlace, se puede siempre volver al nodo origen. Además *Storyspace* guarda un registro de recortes vistos, a la manera de los navegadores de Internet, que se puede recorrer o visualizar.

Toda esta funcionalidad recuerda a un navegador de Internet; sin embargo, el aspecto más interesante, que supera las posibilidades de un navegador, es la coexistencia de varios enlaces. Si de un segmento de texto parten varios enlaces, al hacer clic sobre éste aparece un cuadro de diálogo donde se enumeran los recortes de destino, entre los cuales el usuario puede elegir el que desee. Si varios enlaces parten del nodo completo, aparecen en orden de prioridad, siguiéndose el de prioridad más alta, a no ser que éste tenga atribuido un "campo de prevención" que lo impida.

El "campo de prevención" es un pequeño filtro que permite establecer requisitos simples, como que se haya seleccionado un texto particular, o que otro nodo se haya visitado con anterioridad, o no se haya visitado, antes de poder activar el enlace. Evidentemente, esto es útil para la creación de documentos que poseen un cierto grado de interactividad, e incluso de impredectibilidad.

Aparte de esta navegación automática, existen otras tres formas de utilizar los enlaces. La primera de ellas se basa en que el usuario puede, si así lo desea, dar nombre a un enlace. La ventana de *paths* o recorridos enumera todos los nombres de enlaces y, a continuación, todos los nodos a ambos lados de los enlaces. A partir de aquí, por supuesto, se puede abrir el texto de un nodo. Esta función no interviene en la navegación automática por los enlaces, por lo que, por ejemplo, no existe una regla que haga que se abandone un nodo mediante un enlace que tenga el mismo nombre que el utilizado para entrar. Por este motivo no se puede seguir automáticamente un recorrido como medio de navegar por un documento.

Otro procedimiento consiste en "ojear enlaces", que muestra una ventana con la lista de todos los enlaces que tienen como origen el nodo actual. Esta ventana es la que se utiliza también para cambiar el nombre y las condiciones de activación de un enlace, y para establecer su prioridad (esto último se hace alternando su posición en la lista).

En último lugar, la ventana *Roadmap* proporciona un mapa local que contiene los nodos en que se originan los enlaces que conducen al nodo actual, así como los nodos que son destino de los enlaces que nacen en el nodo actual. Este mapa muestra también el texto del nodo actual y al hacer doble clic en un nodo de cualquiera de las dos listas, convierte a éste en el recorte actual. Es posible abrir varias ventanas de mapa de manera simultánea, lo que permite examinar el documento partiendo de sus enlaces.

Lo peor de la organización de enlaces en *Storyspace* es el tratamiento de los enlaces que apuntan a un segmento de texto particular dentro de otro nodo. A este respecto,

existen dos problemas. En primer lugar, cuando se sigue un enlace hasta un segmento de texto, el segmento no aparece seleccionado, de modo que no se distingue lo que se supone que es relevante en el texto de destino. En segundo lugar, cuando se observa un segmento de texto, no hay manera de saber que existe un enlace que conduce a él, lo que dificulta mucho el mantenimiento del documento y convierte a esta función en impredecible.

Un nodo puede tener cualquier número de palabras clave asociadas. Esto es útil a la hora de realizar búsquedas. En la ventana "Buscar" se puede encontrar nodos tanto si contienen cierto texto como si tienen determinadas palabras clave, o ambas. Desafortunadamente, no es posible hacer búsquedas más complejas, por ejemplo, no es posible buscar nodos que contengan dos palabras clave determinadas.

El otro uso de las palabras clave está en relación con la búsqueda automática de vínculos. Hay dos palabras "mágicas", de forma que, cuando se navega dentro de un nodo que tiene una de ellas, un sub-nodo no visitado dentro de ese nodo aparecerá en su lugar. Hay otra palabra "mágica" que limpia el historial, provocando que todos los nodos aparezcan como "no visitados". Este uso de las palabras clave resulta confuso ya que la capacidad de las palabras clave se ve distorsionada para poder implementar la distinción entre un campo asociado a un vínculo y un campo asociado a un nodo.

Otro aspecto importante a tener en cuenta es la compartición de los documentos creados con *Storyspace*. Una primera opción es que el receptor tenga una copia del programa *Reader Storyspace*. Este programa de distribución gratuita carece de cualquier posibilidad de editar o guardar. Se tiene un cierto control sobre cómo de simplificado será el aspecto del diseño de pantalla que verá el usuario. Por ejemplo, es posible eliminar ventanas de visión para fomentar una navegación a través de los vínculos, o se puede indicar si eligiendo un enlace cierra el nodo de texto principal. También es posible decidir sobre cuál de las dos barras de herramientas verá el usuario. Una de ellas permite libre navegación a través de la jerarquía (ve al nodo hermano de éste, al primer sub-nodo, y así sucesivamente), mientras que la otra obliga a que toda la navegación se haga a través de hacer clic en los nodos, o escribiendo palabras: las palabras tecleadas funcionan como texto seleccionado a efectos de los campos contenedores.

El aspecto del *Reader* se ha mejorado notablemente desde versiones anteriores, en las cuales el usuario tiene acceso a casi todos los menús del propio *Storyspace*. Por ejemplo, el usuario puede hacer una búsqueda para buscar un texto o una palabra clave, puede trabajar con el *Roadmap* e incluso puede abrir la ventana de *Paths*, lo que hace a estas características útiles en un sentido que antes no lo eran. Sin embargo, quizá el usuario tenga demasiado poder: por ejemplo, puede renombrar vínculos utilizando la ventana de *Paths* (aunque este cambio no se puede guardar) y puede limpiar el historial.

La segunda forma de compartir un documento *Storyspace* es exportarlo como HTML. Esto es notablemente sofisticado: cada nodo es una página, se intenta mantener el estilo del texto, las imágenes se exportan y, lo más importante, las plantillas te permiten escribir el HTML en el cual se incrustará el material exportado, e incluir información compleja en una página. Por ejemplo, es posible especificar que, si un nodo es un sub-nodo, incluya un vínculo a su nodo origen, consistiendo en una frase "sube a" y el título

de ese nodo. Las plantillas también permiten compensar al usuario por la pérdida de algunas de las características de *Storyspace* que no pueden trasladarse: por ejemplo, HTML no puede prever un enlace que surja en medio de una página completa, pero es posible especificar que aparezca una lista con todos esos enlaces. Otras características, como los campos contenedores, se pierden por completo.

Finalmente, se puede exportar como texto. Se pierden los estilos e imágenes, pueden exportarse todos los nodos, los nodos actuales y sus sub-nodos, o todos los nodos que pertenezcan a un *Path*. Los ficheros de plantilla permiten personalizar la exportación ligeramente, por ejemplo es posible insertar un separador entre nodos.

No hay duda de que esta es una versión con grandes mejoras de *Storyspace:* la interfaz es a la vez más simple y potente, y por lo tanto mucho más sencilla de manejar.

A modo de resumen se puede decir que en la actualidad, *Storyspace* genera hipertextos que pueden crearse y distribuirse libremente, además de que pueden guardarse como un programa aislado o exportarse a la *World Wide Web.* Permite la creación de estructuras de hipertexto muy ricas. El mapa *Storyspace* muestra cada espacio de escritura del hipertexto y cada uno de sus enlaces. Los escritores pueden añadir, enlazar y reorganizar el hipertexto moviendo los espacios de escritura en el mapa. Además, esta herramienta permite una exploración flexible.

3.4.1.2 Tinderbox™

Tinderbox incorpora la mayor parte de la metáfora básica y la interfaz de *Storyspace*; en apariencia, los dos programas son casi idénticos, pero cada uno de ellos está orientado de una manera diferente. *Storyspace* trata con narrativa hipertextual, por lo que presupone la existencia de un autor y una audiencia, razón por la que usa mecanismos como los "campos de prevención" y el programa *Reader* para dirigir a la audiencia por una narrativa no lineal. Por su parte, *Tinderbox* carece de estos mecanismos, pero introduce otros nuevos: está orientado al usuario individual y su propósito es actuar como una especie de base de datos ligera donde guardar pequeños trozos de texto, como una utilidad que permita tomar notas, organizar los pedazos de información y quizá exportarlos como HTML. En este sentido, *Tinderbox* proporciona funcionalidades de almacenaje, organización y recuperación de la información más interesantes y potentes que las ofrecidas por *Storyspace.*

En *Tinderbox* la unidad básica es el fragmento de texto, al que se denomina nota. Una nota consta de dos partes: su nombre y su contenido real, que puede tener formato y contener imágenes, y que además puede ser editada en la ventana de texto de nota. Una nota puede estar ubicada dentro de otra nota, y las sub-notas dentro de una misma nota tienen su propio orden, que el usuario puede modificar y reorganizar. De esta forma se crea una organización jerárquica entre ellas que puede ser visualizada de varias formas: vista de esquema, vista de gráfica, vista de mapa y vista de árbol. Las notas también pueden relacionarse unas con otras por medio de enlaces. Un enlace puede nacer de una nota completa o de un segmento de texto en el interior de la misma, terminando siempre

en otra nota. Al activar un enlace desde la nota en la que tiene su origen, se abre una ventana que contiene el texto de la nota destino. En último lugar, y al igual que sucedía en *Storyspace*, un enlace puede tener asignado un nombre.

Familiarizarse con *Tinderbox* es bastante sencillo. Se puede comenzar creando y accediendo a nodos sucesivos sin usar el ratón: la tecla de retorno de carro crea una nueva nota, la barra espaciadora abre su ventana de texto, la tecla de inicio de *Windows* (ó *Apple*) –W la cierra y la tecla *Enter* permite cambiarle el nombre. Una vez creadas algunas notas, pueden reorganizarse; la manera más fácil de hacerlo es usando la vista de esquema, donde basta con arrastrarlas o usar accesos rápidos del teclado. Crear enlaces es igualmente simple, debe seleccionarse una nota o un fragmento de texto dentro de una nota, teclear Alt-Windows-L y hacer clic en el destino del enlace. Existen otras formas de realizar estas acciones, pero de esta forma se puede comenzar a trabajar con la herramienta de una forma muy rápida y eficaz.

A este conjunto básico de "trucos", *Tinderbox* añade dos innovaciones importantes: atributos y agentes.

Los atributos constituyen un modo adicional de organización de fragmentos, estando al mismo nivel que la jerarquía a modo de esquema y los enlaces. Un atributo es simplemente un par valor-nombre, donde el valor puede ser un tipo básico como texto, un número o una fecha, por ejemplo "edad:47". Muchos atributos están creados por defecto, como la fuente en la que aparece el título de una nota, pero es posible crear atributos nuevos. De esta forma *Tinderbox* funciona como una pequeña base de datos. Si, por ejemplo, cada nota que representa a una persona tiene su edad como un atributo, es posible encontrar rápidamente a todas las personas mayores de una cierta edad.

Aunque las notas en realidad no son de diferentes tipos, es posible tratarlas como si lo fueran. Se podría tener notas "persona" con un atributo "edad", notas "libro" con el atributo "ISBN", etc. En realidad, cada nota tiene un valor para cada atributo, por lo que en la práctica una "persona" tendrá un "ISBN", pero esto no tiene importancia porque normalmente el usuario no se encontrará con él al trabajar.

Se puede establecer que una nota muestre sus atributos particulares en un panel en la parte de arriba de su ventana de texto. De esta forma, mientras se corrige el texto de una nota "persona" es posible ver su edad en la parte de arriba de la ventana, pero no su "ISBN". Y este "ISBN" tendrá un valor por defecto como el cero o un campo vacío, por lo que ninguna de las búsquedas de "libro" encontrará a una nota "persona".

Existen muchos modos de ver y manipular atributos, como el que se acaba de mencionar de mostrar los atributos en un panel situado en la parte de arriba de la ventana de texto de una nota. Ese panel muestra y permite modificar todos los atributos de una nota en concreto. Un *stamp* es más o menos lo contrario. Consiste en un valor particular para un atributo particular, que puede ser aplicado a todas las notas seleccionadas mediante el uso del menú "Valor" o de la ventana *Quick Stamp*. Un prototipo es una nota que actúa como una plantilla; si otras notas tienen asignada esta nota como su prototipo, heredan todo los valores de sus atributos. Finalmente, un acción es una asignación de

atributos que una nota realiza sobre otra que se convierte en su sub-nota (es decir, en el momento de la creación en su interior o cuando es movida dentro de ella). Esto constituye una característica muy potente que, obviamente, debe ser manejada con cuidado.

Con respecto a los agentes, para comprender el funcionamiento el usuario debe estar familiarizado con el uso de alias. Un alias en *Tinderbox* es un nombre que se le asigna a una nota y que puede ser usado en cualquier parte, permitiendo de esta forma que la misma nota sea representada en distintas posiciones de la jerarquía.

Un agente es una especie de consulta sobre todas las notas del documento. Ahora *Tinderbox* ya tiene una funcionalidad de búsqueda, por lo que cabe preguntarse qué tiene de diferente un agente. La diferencia es que un agente es en sí mismo una nota, con la particularidad de que uno de sus atributos es una consulta. Además, todas las sub-notas de una nota agente son alias de aquellas notas que satisfacen su consulta. Esta forma de búsqueda y agrupación de la información en *Tinderbox* es automática y dinámica. Constantemente se revisa detenidamente el documento y se actualizan las asociaciones llevadas a cabo por cada agente. Por ejemplo, si un agente busca todas las notas que contienen la palabra "mar" y el usuario escribe esa palabra en el texto de una nota, un alias de esta nota aparecerá automáticamente entre las sub-notas de ese agente. Por lo tanto, los agentes proporcionan agrupaciones simultáneas y alternativas de las notas que ayudan a no perder la visión global y actualizada de la información.

Esta herramienta tiene además algunos otros rasgos que merece la pena mencionar. *Storyspace* limitaba los nombres de las notas a 32 caracteres, lo que no era demasiado funcional. *Tinderbox* elimina esta limitación, por lo que ahora las notas pueden tener nombres significativos y se puede usar la vista de esquema con verdadero sentido. Además *Tinderbox* recuerda los nombres de los enlaces a nivel global, por lo que para darle a un enlace un nombre que ya ha sido usado, se puede escoger este nombre de un menú *pop-up* (en vez de tener que recordar y teclear el nombre cada vez como ocurría en *Storyspace*). Los agentes pueden realizar búsquedas sobre los enlaces, por lo que se puede realizar una búsqueda para las notas unidas mediante un enlace "estar de acuerdo", lo que hace que los nombres de los enlaces resulten muy útiles.

Por otro lado, si una palabra en la ventana de texto de una nota tiene capitalización interna (*comoEsta)*, entonces si se hace *Windows-Alt*-clic en esa palabra (combinación que se usa normalmente para seguir un enlace), pero no existe ningún enlace, *Tinderbox* intentará trabajar con esa palabra como si realmente el enlace existiera. Esto significa que si esa palabra era el nombre de una nota, abrirá el cuadro de texto de esa nota, y si no lo era, ofrecerá la posibilidad de crear una nota con ese nombre.

Una nota puede tener asociado un archivo simplemente arrastrando el archivo al icono de archivo de la ventana de texto, donde un menú permite al usuario abrir dicho archivo. Por lo tanto, *Tinderbox* puede ser usado como un interfaz de organización de archivos en disco. Además, las sub-notas pueden ser clasificadas conforme a criterios especificados en los atributos de la nota a la que pertenecen.

Tinderbox incorpora además una nueva vista muy conveniente, la vista de explorador, que presenta a la izquierda las notas en una lista, una gráfica o un mapa y a la derecha muestra el texto de la nota seleccionada en el panel de la izquierda. Esta herramienta tiene también un número de funcionalidades orientadas a Internet, por ejemplo, un enlace del texto puede ser ahora un enlace a una página web. Además, *Tinderbox* es un cliente web: una nota puede tener un atributo "URL" y su texto será el texto de esa URL, descargado bajo demanda. Sin embargo, *Tinderbox* no es un navegador, por lo que si el texto es HTML, simplemente puede, o bien mostrar el texto HTML, o bien dejar que un navegador abra y muestre la página.

El usuario puede también usar *Tinderbox* para exportar notas como HTML usando una plantilla. Una plantilla consiste en un fichero de texto HTML con espacio para los elementos que deben provenir de cada nota. Los enlaces entre texto de una nota y otra nota se conservan como enlaces HTML, la estructura jerárquica del documento se conserva, y además es posible especificar enlaces de navegación para favorecer la movilidad del usuario por la misma. El mecanismo de las plantillas es simple pero bastante potente, por ejemplo, permite crear elementos de plantilla condicionales. Además, los detalles acerca de cómo será exportada cada nota en concreto se establecen en sus atributos. Esto quiere decir que, por ejemplo, todas las notas podrían usar una cierta plantilla por defecto, pero algunas notas particulares podrían usar otra plantilla diferente. La exportación de una nota puede incluir la exportación de sus sub-notas, y por supuesto, una plantilla puede tener acceso a cualquier atributo de una nota, combinando así las funcionalidades de una base de datos ligera con características de exportación HTML.

Todas estas características de exportación HTML pueden ser usadas para crear sitios web. El manual describe, por ejemplo, la posibilidad de exportar un agente junto con todas sus sub-notas. Si la consulta del agente es para notas creadas en las últimas dos semanas, clasificadas por su fecha de creación, lo que se obtiene es un blog (ya han aparecido varios blogs creados usando *Tinderbox)*.

Sin embargo, a pesar de todas estas características, esta herramienta presenta algunas carencias en su funcionamiento. Por ejemplo, cuando se cambia la consulta de un agente utilizando *Quick Stamp*, los resultados de la búsqueda del agente no se actualizan. Por otro lado, un alias tiene acceso al texto y los atributos de su original, pero no muestra sus sub-notas, y en último lugar, creo que sería muy deseable que la exportación de texto no estuviera limitada a texto plano, sino que permitiera dar estilo al texto, de esta forma *Tinderbox* se convertiría en una verdadera herramienta de escritura.

3.4.1.3 Evaluación de las herramientas analizadas

En apartados anteriores se han analizado en profundidad todas las funcionalidades y características que ofrecen a sus usuarios dos de las herramientas de escritura más utilizadas por autores de narrativa hipertextual. Este análisis detallado de ambas herramientas viene motivado por la situación actual en la que se encuentra este tipo de

narrativa, ya que el principal problema al que tiene que enfrentarse es la falta de integración entre la teoría y la práctica.

Este pequeño estudio de campo permite poner de relieve los "puntos débiles" de las herramientas disponibles actualmente de manera que estas carencias puedan tratar de evitarse en la aplicación que se va a desarrollar. Algunas de estas debilidades se exponen en los párrafos siguientes. Sin embargo, el gran reto de HyperAuthor no consiste en superar a las herramientas actuales, sino en plantear como base y llevar a la práctica una filosofía completamente diferente que traiga consigo una manera distinta de concebir la hiperficción.

Un aspecto importante a mencionar del análisis previo es que las herramientas están diseñadas para usuarios pertenecientes al mundo "técnico" a los que se les presupone unos conocimientos mínimos. Esto es claramente un limitante para el mundo de las letras y las humanidades, quienes serán los principales usuarios de este tipo de herramientas de autor.

Finalmente, ambas ofrecen una integración de "medios" limitada para hacer posible la literatura hipertextual, además de estar diseñadas en su mayoría para trabajar en Mac (y no en Windows, aunque van apareciendo adaptaciones).

Debido a estas limitaciones, como ya se ha comentado en apartados anteriores, muchas de las obras que se encuentran actualmente en Internet y que pretenden ser narrativa hipertextual no son más que versiones mejoradas de las notas al pie de página, resultando en obras que preservan la mayoría de los elementos de la ficción lineal como son el argumento, los personajes, etc.

De esta forma, esta nueva forma de narrar se ve forzada a continuar dependiendo del género del que nació, lo que supone una barrera importante para la divulgación, y sobre todo para la evolución, de este incipiente género.

3.4.2 Principios de HyperAuthor: una herramienta integradora

Los teóricos del hipertexto han propuesto una serie de puntos o características deseables en cualquier obra hipertextual, pero para llevarlas a la práctica existen pocos modelos y herramientas que sirvan de orientación a los autores que intentan adentrarse en esta nueva forma de narrativa.

Encontramos, por lo tanto, una carencia de herramientas o modelos adecuados para el escenario actual. Esta es la razón que motiva la creación de nuestro modelo, cuya finalidad es la de proponer una forma inicial de llevar a la práctica algunos de los postulados teóricos propuestos. Por otra parte, técnicamente este modelo servirá como base para la creación de una herramienta completa y adecuada en la que los escritores puedan dar sus primeros pasos con las nuevas formas de narrativa.

3.4.2.1 Filosofía del modelo

El modelo base que se va a usar para desarrollar nuestra herramienta integradora se expone en el apartado siguiente, pero antes de entrar en materia se expone a continuación la filosofía sobre la que se sustenta, ya que es ésta filosofía la que lo diferencia de las herramientas existentes en el mercado actual.

Para explicar la filosofía subyacente en este modelo se ha tomado como metáfora el modelo OITP propuesto por Sáez Vacas en [45] y usado, de forma general, en el estudio de los procesos de negocio en un entorno empresarial. Al adaptar este modelo al escenario en el que se está desarrollando este estudio, el nombre se traduce por el de modelo ERCEN, que nemotécnicamente se expresa por los elementos que aparecen en los vértices del triángulo que lo representa.

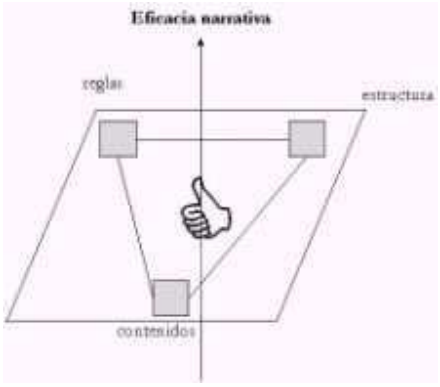


Figura 3.3 Eficacia narrativa.

- La E (estructuras): Representa las estructuras hipertextuales existentes.
- La R (reglas): Se refiere a las restricciones u opciones que el escritor podrá ofrecer al lector en cada nodo de su historia.
- La C (contenidos): Hace referencia a la integración de toda la riqueza de los elementos multimedia.
- La EN (eficacia narrativa): Indica la actividad de transmitir al usuario una experiencia de inmersión en la obra, navegación transparente, comunicarle de modo efectivo la estructura (evitando la sensación de pérdida), responder a sus acciones y ayudarle a explorar la obra.

La separación geométrica entre los vértices intenta visualizar simbólicamente las diferencias en sus relaciones. La idea que la figura trata de representar es que para lograr una auténtica eficacia narrativa es necesario conseguir que los tres vértices del triángulo (sin olvidar ninguno) se aproximen mutuamente, de tal forma que en un tiempo razonable lleguen a la convergencia. El instante en el que los vértices se fundan en un único punto representativo de su armonización en la tecnología, la narrativa alcanzará un nuevo estado en el que la tecnología se integrará plenamente en el proceso creativo.

El vector EN (eficacia narrativa) denota que tanto la elección de la tecnología como las medidas encaminadas al acercamiento de los vértices deben orientarse según alguna línea concreta que pasa por la potenciación instrumental de elementos en y para lograr nuevas formas narrativas.

3.4.2.2 Descripción funcional del modelo

Habiendo explicado las características abstractas y conceptuales en las que se basa el modelo, el siguiente paso es traducirlo a un modelo funcional que logre la sinergia de sus elementos. Este modelo persigue el fin de potenciar el proceso narrativo mediante la utilización de estructuras complejas, contenidos ricos en elementos multimedia y teniendo como nexo de unión un conjunto de reglas que permitan desarrollar toda la creatividad de los escritores, el fin de lograr lo que se ha denominado eficacia narrativa.

El modelo funcional propuesto está representado en la Figura 3.3. Este modelo permite incorporar a la literatura aquellos elementos novedosos que el hipertexto es capaz de aportar, manteniendo al mimo tiempo aquellos aspectos fundamentales de la literatura tradicional; esto es el tiempo profundo y la introspección. El modelo posibilita, por lo tanto, proceder durante el proceso de investigación en materia de hiperficción mediante saltos cualitativos mínimos a partir del libro tradicional. De esta manera se pretende conseguir que el resultado final vaya tan sólo "un poco más allá" y diste cualitativamente poco de este tipo de literatura, ya asentada y madura.

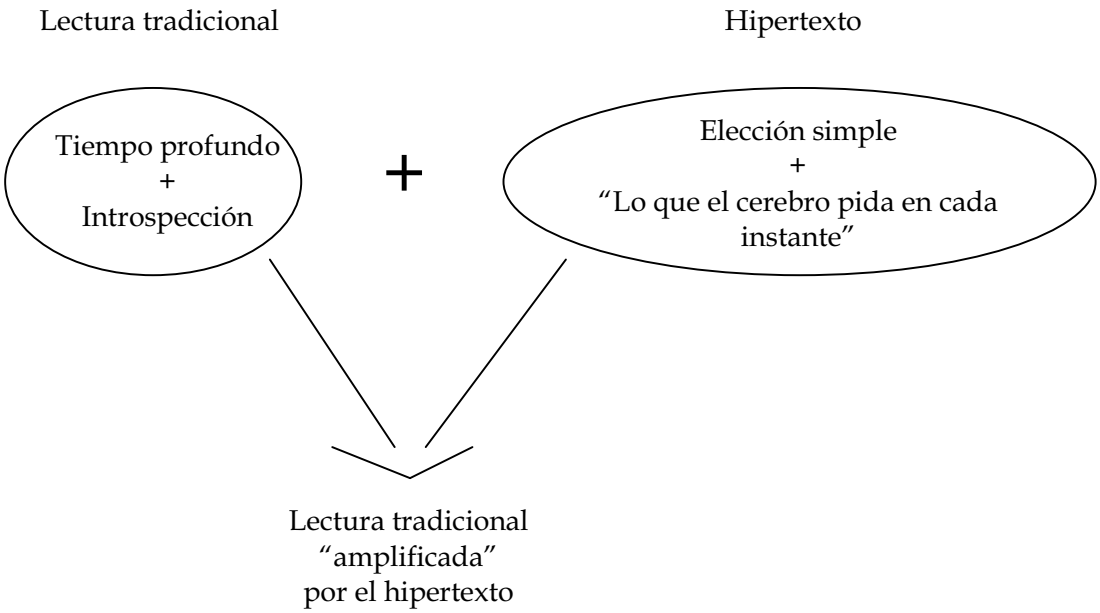


Figura 4.3 Modelo funcional

La lectura de un buen lector, nunca es pasiva, aunque los libros no sean interactivos o inmersivos. Cuando se trata de literatura con mayúsculas, no de literatura-consumo o de literatura-ocio, la lectura siempre es profunda. La hiperficción debe ser capaz de favorecer esa profundidad.

Con esta finalidad, cualquier elemento multimedia o de decisión incluido en el texto debe aumentar la fusión entre relato y lector y apoyar el tiempo profundo. El lector navega por el relato y percibe sus contenidos. En su paso por los nodos va tomando decisiones que han sido previamente establecidas por el autor y que, combinadas con los contenidos, deben propiciar la inmersión del lector en la obra. No es necesario que el relato sea variable, o que se adapte a los deseos del usuario; basta con que el lector lo vaya descubriendo de la misma forma en la que lo hacía con un libro tradicional.

El tema de la inmersión en el relato y la profundidad de la lectura es estudiado en detalle por Birkerts en [4], y sus ideas al respecto deben estar siempre muy presentes durante el desarrollo de la herramienta. Murray se interesa también por este rasgo fundamental de la lectura, y en [17] analiza lo que denomina los tres vectores (transformación, inmersión y actuación), concluyendo que no necesariamente favorecen la profundidad en el relato. Puede ocurrir incluso todo lo contrario, y que la unión de los tres vectores favorezca una profundidad engañosa: el lector puede creerse implicado en la historia por el hecho de que participa de ella y, sin embargo, no está íntimamente ligado a lo que hay sustentando el relato sino únicamente a la participación que éste le ofrece.

La diferencia entre la profundidad real y una ficticia es análoga al "hacer" frente al "pensar" y puede explicarse con una metáfora extraída de la vida cotidiana. Los "maquinistas" se mueven y actúan en la superficie del mundo, participan de los acontecimientos que allí ocurren. Por su parte, los "excavadores" ven en el mundo un reflejo de su propio ser, buscándose a sí mismos al contemplar desde su propia perspectiva interna todo lo que les rodea. La literatura debe estar ligada a lo segundo, mientras que la interactividad fuerte en sistemas narrativos está asociada normalmente con lo primero.

La profundidad en el relato debe, por lo tanto, provenir por el segundo de los caminos. Una obra literaria de calidad deber ser capaz de crear un mundo en el interior del lector, y no de recrearlo fuera de él. "Vivir" y actuar en mundo externo, aunque sea virtual, se convierte en juego o realidad, pero no se trata en ningún caso de literatura. Ésta no pretende (o no debe pretender) la simulación de realidades, por fantasiosas o atrayentes que éstas puedan resultar para el usuario. La finalidad última de la literatura y lo que le da sentido a su existencia es su forma de evocar y reflejar aquello que el lector ya tiene dentro, para después mostrárselo.

En definitiva, y a modo de resumen muy escueto, este modelo pretende mantener el tiempo profundo característico y fundamental de la literatura tradicional y añadir esa componente nueva e interactiva sustentada sobre "da al cerebro lo que pida en cada momento" que el hipertexto es capaz de aportar. De esta forma se consigue potenciar el

proceso narrativo, ofreciéndole al usuario o lector una experiencia de inmersión en la obra, y una navegación transparente que responda a sus acciones. Se consigue, por lo tanto, una literatura tradicional "amplificada" cuya calidad y concepción no se separa demasiado de la literatura convencional.

Pues bien, con estas ideas en mente se enfocó la construcción de HyperAuthor. Hasta aquí ha llegado la exposición del contexto en el que surge la herramienta y los principios que la rigen. En el capítulo siguiente se describe más al detalle el proceso de diseño e implementación de HyperAuthor.

3.5. Conclusiones

Cada uno de los ejemplos tratados en los apartados anteriores demuestra la sinergia existente entre tecnología y literatura y ponen de manifiesto el estado en el que nos encontramos actualmente.

La hiperficción es un género algo más maduro que el resto de los mencionados. De hecho, a pesar de que muchos sitúan sus orígenes en el nacimiento de la World Wide Web, tan extendida y popularizada en nuestros días, ya se trabajaba en hiperficción con anterioridad usando otros formatos (por ejemplo el programa Hypercard). Sin embargo, aunque en Internet es evidente que el uso de hiperenlaces es generalizado y abundante, no es un entorno natural de lectura. No basta con el uso de hipertexto para conseguir hiperficción de calidad, hace falta tener en cuenta consideraciones adicionales. Por ejemplo, hay que cuidar de no tener texto demasiado largo sin ninguna ramificación porque la lectura en una pantalla se hace muy complicada, pero tampoco se puede colocar demasiados enlaces como para que se rompa la línea argumental del relato tan frecuentemente que el lector no sea capaz de sentir la sensación de inmersión en la obra.

El hiperdrama es probablemente, de entre todas las realidades expuestas, aquella que más se aproxima a esa clase de literatura de calidad que puede ofrecer el hipertexto, sin embargo se aleja del cometido de nuestra herramienta en el sentido de que una obra dramática está pensada para ser representada en el teatro y no para su lectura. La popularización de este nuevo género puede y debe ayudar a que el público general se familiarice con una nueva forma de contar historias, pero no tiene que enfrentarse a un tema tan delicado como la lectura en pantalla.

En cuanto a la plataforma descrita en el apartado 2 de este capítulo, se centra no sólo en la generación de una literatura de calidad que aproveche todo el potencial que el hipertexto nos ofrece en la actualidad, sino que se trata de una plataforma pensada de alguna manera para la enseñanza y en la que se da gran relevancia al componente lúdico de la misma para facilitar el acercamiento de los estudiantes a las nuevas tecnologías.

Por otro lado, el tercer apartado poco o nada tiene que ver con el hipertexto, pero sí con el hecho incuestionable de que la literatura y la tecnología están cada vez más íntimamente unidas y por lo tanto, es sólo cuestión de tiempo el que no seamos capaces de entender la una sin la otra.

Todas estas consideraciones nos hacen pensar que existe una carencia clara de herramientas en el mercado que favorezcan la generación de narrativa hipertextual de calidad, que permitan mantener los logros alcanzados por la literatura tradicional y que además den la posibilidad a los autores pioneros que se aventuren a probar suerte en este nuevo género de aprovechar fácilmente algunas de las nuevas características del hipertexto. Esto permitirá que los lectores se acostumbren a este tipo de obras y se pueda avanzar con mayor velocidad hacia su esplendor. Es aquí donde cobra importancia el modelo propuesto como base para el desarrollo de HyperAuthor, cuyos fundamentos han quedado expuestos en el apartado 4 y cuyos procesos de diseño e implementación pueden encontrarse en los capítulos que siguen.

Capítulo 4:
Diseño

4.1 Introducción

Llegados a este punto, parece un buen momento para ahondar en los detalles de la aplicación. En este apartado describiremos el proceso de diseño de la misma. Se parte de los objetivos que se pretenden conseguir y se describen las consideraciones y razonamientos que se fueron sucediendo, en cada caso, hasta llegar a las decisiones finales que se adoptaron para la posterior implementación.

Desde el primer momento en el que se decidió abordar este proyecto, siempre se ha hablado de dos "herramientas" bien diferenciadas, la herramienta creadora y la herramienta para la visualización. La primera, HyperAuthor, debe encargarse de permitir la construcción de la historia hipertextual en sus dos planos y la generación de un documento a partir de la misma. La herramienta de visualización, HyperViewer, debe ser capaz de entender el documento exportado de la herramienta creadora para mostrarlo al usuario y permitir una navegación fácil por el mismo.

Cada una de las dos herramientas ha seguido un proceso de diseño independiente, por lo que es necesario detallar las consideraciones de diseño propias de cada una de ellas de una forma separada. Sin embargo, existen una serie de aspectos que ambas comparten y que se exponen a continuación de una forma común.

La primera de las similitudes, y la más importante de todas ellas, es que ambas herramientas han sido concebidas como instrumento para acercar, tanto a los autores como a los consumidores de literatura, las características específicas de un nuevo tipo de narrativa. De este objetivo común se pueden obtener las primeras consideraciones generales sobre el diseño de las herramientas.

En primer lugar, es importante conseguir hacer llegar este nuevo tipo de literatura a los usuarios finales de las herramientas de una manera que les resulte atractiva. Por lo tanto, en el proceso de diseño de ambas interfaces se ha tenido muy presente qué esta debe resultar lo más colorida y atrayente posible y se ha dado mucha importancia a la accesibilidad de las funcionalidades. De esta manera se evita que el uso de las herramientas resulte tedioso o aburrido.

Por otro lado, no es suficiente con que la literatura hipertextual llegue al gran público, es también necesario que el público se familiarice con ella y la asuma como propia. Por este motivo, uno de los puntos más importantes a tener en cuenta a la hora de adoptar decisiones concretas de diseño ha sido el dotar a las herramientas de una componente claramente didáctica. Es de vital importancia que, una vez superada la barrera de enfrentarse a una herramienta desconocida, el usuario sea capaz, mediante el uso de la misma, de "aprender" a escribir o leer de forma hipertextual.

A este respecto, es importante no olvidar la importancia de la interactividad con el usuario. Ésta es una de las características básicas del hipertexto a la vez que uno de sus mayores atractivos. Como consecuencia, se ha hecho un esfuerzo en el diseño de las herramientas para que ambas contaran con un grado de interactividad lo suficientemente

alto como para que el usuario tenga una sensación de inmersión en el "mundo del hipertexto". De esta manera, las sensaciones experimentadas mediante el uso de las herramientas pretenden ser un reflejo de la experiencia del lector frente a una obra hipertextual.

Todas estas características ponen de relieve la importancia que la interfaz gráfica tiene, ya que todos los principios y novedades que el hipertexto aporta en este caso deben verse reflejados, tanto en HyperAuthor como en HyperViewer, de una forma clara y sencilla. Como consecuencia, el proceso de diseño de las mismas debe ser muy meticuloso, y se ha considerado interesante separarlo del diseño de las funcionalidades internas de las herramientas, motivo por el que se ha adoptado el paradigma Modelo-Vista-Controlador para su implementación.

4.2 HyperAuthor

Hasta aquí se han comentado las directrices generales de diseño que tienen en común HyperAuthor e HyperViewer, por lo que ya sólo resta comentar las particularidades de cada una de las herramientas. Sin más dilación, por lo tanto, se comienza a continuación la exposición detallada de todo lo concerniente a HyperAuthor.

4.2.1 Requisitos

Para empezar, vamos a hacer una breve recapitulación de lo que se pretende lograr con HyperAuthor. El propósito de esta aplicación es proporcionar una herramienta para la creación de narrativa hipertextual teniendo siempre presente dos premisas que consideramos básicas: las obras creadas con esta herramienta no deben perder los logros alcanzados por la narrativa tradicional y deben además incorporar algunas características nuevas (gracias al hipertexto) que ayuden a mejorar y aumentar la unión entre la obra y el lector.

Es decir, se enfoca la hiperficción no como una ruptura total con la literatura tradicional, sino como una generalización de la misma, en la que la literatura tradicional está incluida. En nuestro caso, además, el paso dado desde la literatura tradicional hacia la hiperficción no será grande, sino todo lo contrario. Esto permite controlar que no nos alejaremos en exceso de la primera y que, por lo tanto, las obras creadas contendrán aquellas características que es necesario mantener y perpetuar.

Con HyperAuthor, se pretende hacer hincapié en la importancia que toma el plano estructural en una obra hipertextual y que hace que conceptos que eran fundamentales en literatura lineal, como lo era el argumento, pasen a un segundo plano. Se pretende poner de relieve que, en literatura hipertextual, la estructura del relato es uno de los elementos más importantes a tener en cuenta cuando un autor se sienta a escribir, y que su diseño debe abordarse de forma separada y previa al del contenido de la obra en sí.

La finalidad que se persigue es, por tanto, proporcionar a los autores una herramienta con la que puedan "aprender" las estructuras básicas existentes para la creación de hiperficción, la semántica de tales estructuras y el concepto que subyace en ellas, el modo de usarlas… De forma más general, lo que se pretende conseguir es enseñar a los autores que se adentran en el mundo del hipertexto las nuevas características que introduce la literatura hipertextual así como a aplicarlas de forma adecuada en la creación de nuevas obras.

Con estas premisas, y teniendo en cuenta las consideraciones expuestas en capítulos anteriores (ver capítulo 2), la aplicación parte de los siguientes requisitos:

- Proporcionar un entorno en el que el autor entre en contacto y pueda operar directamente con las estructuras básicas de la hiperficción construyendo a partir de ellas nuevas obras.

- Ofrecer en la herramienta dos planos de diseño distintos y bien diferenciados: el plano de contenido que hace referencia al texto en sí y con el que los autores están más familiarizados, y el plano estructural, más desconocido para ellos y que se encuentra asociado a la forma de la narración.

- Mantener un enfoque claramente atrayente, simple e intuitivo. Por este motivo, la construcción de la estructura de las nuevas obras se llevará a cabo de manera gráfica y para la edición de nodos se utilizará un editor de texto adaptado muy similar a los ya existentes y que incluya las nuevas funcionalidades, como es la interconexión de nodos creados.

- Permitir al autor navegar por el hipertexto creado desde la propia herramienta de creación. Aunque este modo no ofrece todas las funcionalidades de HyperViewer, le permite al autor aproximarse a las sensaciones del lector ante su obra.

- Construir, adicionalmente a la herramienta creadora, un visor de los documentos que incluya funcionalidades como la creación de un histórico de navegación.

4.2.2 Planteamiento general

Comenzamos por lo concerniente a HyperAuthor. Decidimos aplicar al diseño de nuestra herramienta el patrón Modelo-Vista-Controlador (MVC) [28], un patrón pensado para separar la capa de presentación de la "lógica de negocio". Se trata de un patrón en el que se manejan tres tipos de módulos:

- Vista: Compuesto esencialmente por la interfaz gráfica que, como es bien sabido, es la encargada de interactuar con el usuario. En nuestro caso, se

encarga, además del tratamiento de los elementos gráficos como botones o menús, de la representación y mantenimiento en pantalla de los grafos que describen la estructura de la obra.

- Controlador: Es el eslabón que une la vista y el modelo. Se encarga de recibir los eventos de entrada así como de gestionarlos. La respuesta a estos eventos pueden suponer peticiones al modelo o a la vista. En nuestro caso, parte de la gestión de eventos está integrada en la vista.

- Modelo: Se encarga del acceso a los datos y de la implementación de toda la funcionalidad de la herramienta. Por simplicidad, se decidió que este módulo se encargase también del almacenamiento de los documentos generados en discos (persistencia) y su recuperación.

La finalidad básica del patrón MVC es la de desacoplar la vista de la aplicación del modelo en sí, con la finalidad de mejorar su reusabilidad. De esta forma, las modificaciones en la vista tienen un menor impacto en la lógica de negocio o de datos. Esto es siempre una buena idea para cualquier tipo de aplicación con interfaz de usuario, pero aún más en nuestros caso, y vamos a ver el porqué.

HyperAuthor nace con la idea de representar sólo un pequeño paso adelante desde la literatura tradicional hacia la hiperficción, por lo que, a pesar de que la herramienta es capaz de abrir un nuevo mundo a los autores que se adentran en este género, sus funcionalidades están limitadas de modo que el resultado no sólo no sea demasiado desconcertante, sino que además sirva para acercar al gran público este nuevo tipo de narrativa. Por lo tanto, a medida que los usuarios se habitúan a la narrativa hipertextual, la herramienta creada debe ser capaz también de ir evolucionando e introducir cada vez más funcionalidades, de manera que se llegue al aprovechamiento total de las posibilidades del hipertexto mediante pequeños pasos independientes.

Esto implica que la herramienta debe permanecer en evolución de un modo casi constante, por lo que es importante que exista el desacoplo entre lógica e interfaz de usuario para facilitar esta tarea evolutiva.

Sin embargo, a la hora de llevar a cabo la implementación real de la herramienta nos encontramos con una pequeña dificultad, y es que es complicado desacoplar totalmente el módulo de vista del controlador. Este es el motivo por el que, en nuestro caso, parte del procesamiento de eventos queda incluido en clases de la vista, de modo que el controlador queda semioculto dentro de la misma. A pesar de esto, se ha conseguido una aproximación bastante buena al patrón.

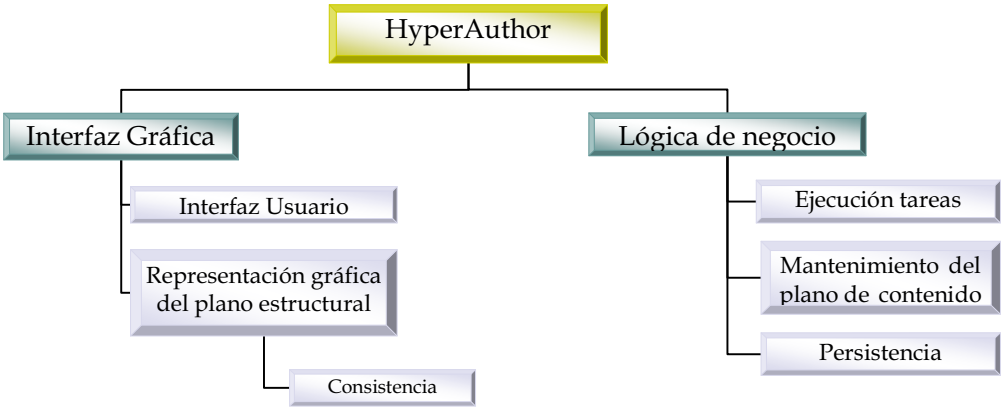


Figura 4.5 Arquitectura de HyperAuthor.

En los apartados siguientes de este capítulo se va a hacer un análisis en detalle de cada uno de los bloques funcionales en los que se ha dividido la herramienta; ahora, lo importante es justificar esta división.

Ya se ha comentado con anterioridad que HyperAuthor pretende dividir el diseño de una obra hipertextual en dos planos separados: el estructural y el de contenido. El plano de contenido hacer referencia tanto al texto como al resto de elementos hipermedia que conforman la propia historia. En este plano el usuario trata con un editor de texto similar a aquellos con los que ya está familiarizado. Sin embargo, el plano estructural ofrece al usuario un editor exclusivamente gráfico, de forma que para construir la estructura completa de una determinada obra, se entienda el uso de los elementos gráficos que ofrece la herramienta como una forma distinta de escritura. Una forma distinta a la que empleaban hasta ahora, pero que al igual que cualquier otra, posee su propia "gramática", es decir, existen ciertas reglas impuestas por la herramienta y que la representación gráfica debe cumplir.

Por este motivo debe existir una cierta restricción de las funciones que se pueden realizar dependiendo del estado en que se encuentre la representación en pantalla además de un mantenimiento y validación de la consistencia de la misma después de cada acción invocada por el usuario.

Anteriormente se ha expuesto que la interfaz gráfica adquiere también esta responsabilidad de representación y mantenimiento de los grafos que muestran la estructura hipertextual en pantalla. Esta labor requiere una proyección constante en la interfaz de usuario y una interacción constante con la misma. El usuario va a manejar objetos gráficos que deben poder insertarse, borrarse, interconectarse, copiarse, etc. Este proceso de "escritura en pantalla" debe ajustarse a ciertas restricciones impuestas por la herramienta.

Desde la perspectiva de la orientación a objetos, se decidió que sería una solución más compacta añadir a los objetos gráficos la inteligencia necesaria que permitiese implementar las restricciones de dibujo que íbamos a necesitar imponer, es decir, las restricciones concretas de nuestra herramienta que hacen que un determinado grafo sea consistente en todo momento.

Resultaba mucho más adecuado entonces incluir la representación gráfica y el mantenimiento de la estructura hipertextual al mismo nivel que el resto de las funciones gráficas, de forma que se evitara la participación de clases distintas a las ya involucradas en procesos gráficos en esta tarea.

Teniendo en consideración todo lo mencionado hasta el momento, la aplicación queda a grandes rasgos divida en dos grandes bloques:

- Interfaz gráfica: el conjunto de funcionalidades de índole gráfica y representación y mantenimiento de la estructura hipertextual en pantalla. También incluimos en este bloque parte de la gestión de eventos relacionados con la interfaz.

- Lógica de negocio: incluye todos los procesos y funcionalidades internas de la aplicación.

De este modo, la interfaz gráfica de HyperAuthor pasa a ser un concepto más amplio que constaría a su vez de los dos grandes bloques que se muestran en la figura 4.1

- Interfaz de usuario: que incluye aquello que participa en la interacción con el usuario: botones, menús, diálogos, etc. así como la gestión de eventos.

- Generación gráfica de la estructura hipertextual: que incluye las funciones gráficas que permiten dibujar los diagramas en pantallas y garantizar su corrección.

Antes de pasar a explicar cada bloque con más detalle, vamos a establecer algunos convenios de nomenclatura que emplearemos en adelante. Hasta ahora se ha hablado de que el usuario, valiéndose de HyperAuthor, va a crear obras hipertextuales. Pues bien, la obra para nosotros se referirá a un concepto abstracto de aquello que el autor está creando. Sin embargo, dada la separación que se ha hecho de los dos planos diferentes a tener en cuenta para la creación de una obra, se hablará de:

- Grafo o diagrama: representa la obra en el plano estructural a nivel de interfaz gráfica.

- Celda: cada uno de los elementos que conforman el grafo. Pueden ser nodos o ejes.

- Nodo: cada uno de los elementos de contenido que pertenecen al grafo. Estos nodos almacenan el texto y demás elementos introducidos por el autor en el plano de contenido, por lo que enlazan con el mismo.

- Eje: cada uno de los elementos de enlace que forman parte del grafo a nivel de interfaz gráfica. Conecta dos nodos.

- Enlaces: cada uno de los elementos de enlace a nivel de contenido. Se definen por el autor en el editor de textos y tienen una correspondencia unívoca con un eje en el grafo.

- Árbol: representación del grafo cuando es manejada por la lógica de negocio.

Ahora, sin más dilación, en los siguientes apartados se va a analizar cada uno de los bloques.

4.2.3 La interfaz gráfica

Ya se ha expuesto que, de acuerdo con los objetivos asumidos en el desarrollo de esta aplicación, y dado el enfoque mediante el cual pretendemos conseguirlos (descritos ambos en los capítulos anteriores) la interfaz de usuario de HyperAuthor resulta ser una pieza muy importante en este proyecto.

En todas las aplicaciones la interfaz de usuario determina la "usabilidad" del programa, el grado de aprovechamiento de las funcionalidades que ofrece y, por supuesto gran parte de la opinión del usuario sobre el mismo, motivos por los cuales, indirectamente el éxito o fracaso de una aplicación depende en gran medida de su interfaz exterior. En HyperAuthor, la interfaz debe cumplir además una serie de objetivos adicionales:

A. Construcción gráfica de la estructura.
La construcción de la estructura del hipertexto (dimensión estructural) debe ser un proceso completamente gráfico. El atractivo que, como se ha comentado a lo largo de este documento, se pretende aportar a esta aplicación, descansa fundamentalmente en este punto.

B. Acceso rápido a las estructuras.
El acceso a estas estructuras debe ser inmediato y sencillo, de modo que el proceso de su utilización supusiese la menor carga cognitiva para el usuario.

C. Interfaz gráfica familiar.
A pesar de ser una herramienta específica, su apariencia debe guardar la mayor semejanza posible con cualquier aplicación convencional de tratamiento de texto (ya que los autores están familiarizados con este tipo

de aplicaciones). De este modo, la curva de aprendizaje se reduce considerablemente y, lo que es aún más importante, los autores son capaces de centrar su atención desde el primer momento en el aprendizaje de la nueva forma de escritura a la que se enfrentan.

D. Representación gráfica de las estructuras.
Proporcionar una representación adecuada de las estructuras tipo básicas que permite utilizar la herramienta para la construcción del hipertexto. Éstas son las piezas básicas del plano estructural con el que el autor debe familiarizarse y aprender a usar con corrección.

E. Conexión de las dos dimensiones de diseño.
La interfaz debe implementar mecanismos para conectar la dimensión estructural con la dimensión de contenido, de modo que el usuario pueda interactuar con las dos dimensiones de la herramienta a partir de una misma interfaz.

F. Ayudas a la navegación.
Se deben construir navegadores y mecanismos de ayuda a la navegación que permitan al usuario de la herramienta creadora moverse por la red de hipertexto mediante la activación de los enlaces insertados en el texto.

G. Personalización de la herramienta.
Por último, se consideró interesante incluir en la propia interfaz de la herramienta algunas opciones de personalización de la misma para facilitar el trabajo de una manera cómoda por parte del autor.

Cada uno de los puntos anteriores ha hecho necesario la toma de una serie de decisiones que se detallan, por orden, a continuación.

A. Construcción gráfica de la estructura.

En consonancia con lo que ya se ha comentado reiteradamente en este documento, se pretende que el proceso de construcción de la estructura hipertextual se realice de forma completamente gráfica e interactiva. Como se señalaba en el apartado 4.1.2.2, este requisito iba a necesitar de cierta capacidad de procesamiento por parte de la interfaz gráfica.

La idea a este respecto es que las obras que se diseñan usando HyperAuthor se "construyen" dibujando un grafo que represente su estructura en el área central de trabajo de la aplicación. Para crear este diagrama, las estructuras básicas serían introducidas e integradas en el área de trabajo mediante el uso de la barra de herramientas situada a la izquierda de la pantalla.

Esto crea la necesidad de modelar la estructura de una obra como un objeto gráfico, denominado grafo, que está a su vez compuesto por otros objetos gráficos como son los nodos y los ejes que los interconectan y representan la forma de lectura.

Sin embargo, y como consecuencia de que HyperAuthor pretende ser simplemente un pequeño paso adelante en el camino a recorrer desde la literatura tradicional hasta la hipertextual, no cualquier combinación de nodos y las conexiones entre ellos están permitidas en la herramienta. En la construcción y manipulación del grafo se deben imponer ciertas limitaciones para que la obra resultante resulte atractiva para los lectores y conserve aquellas características de la literatura impresa necesarias para crear hiperficción de calidad; esto implica definir un modelo de grafo.

El modelo de un grafo es el conjunto de reglas que definen de algún modo la forma que tendrá el grafo, o bien, también puede ser definido como el conjunto de limitaciones que lo separan de un grafo genérico. Llegados a este punto, entramos en lo que constituye el núcleo duro del diseño de la herramienta en la dimensión estructural (que es la que maneja la interfaz gráfica): el proceso de construcción del hipertexto.

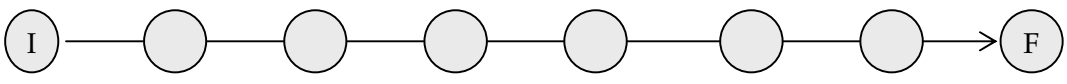
Para abordar este aspecto hay que centrarse en los dos problemas básicos que puede presentar un hipertexto mal diseñado: la sensación de desorientación que puede experimentar el usuario al navegar por una red hipertextual y la impresión de "estar perdiéndose cosas" que normalmente sienten los lectores ante la toma de una decisión que supone el elegir un camino (representado por un enlace) descartando otros.

En lo referente al problema de desorientación, se pensó que una buena forma de evitarlo era trabajar siempre con estructuras definidas (lo que se ha estado denominando desde el principio como estructura tipo o estructuras básicas) en lugar de con nodos sueltos. Por este motivo, la estructura hipertextual se construye a partir de estas secuencias bien definidas que se introducen siempre en relación con un nodo que constituye su origen y final.

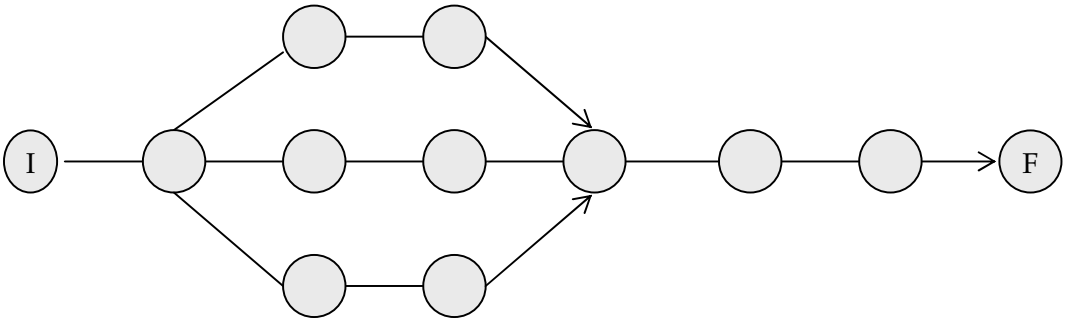
Se descartó, por lo tanto, la posibilidad de introducir nodos sueltos en cualquier parte del área de trabajo y enlazarlos luego con la estructura existente mediante el dibujo de ejes, ya que de esta forma no se podía controlar la forma global de la estructura. Únicamente se permite la introducción de nodos sueltos para alargar una secuencia ya existente, y estos nodos se introducen dentro de la misma tomando como referencia de nuevo el nodo seleccionado.

En este sentido, se analizaron varios hipertextos tratando de descomponerlos en aquellas estructuras o secuencias más simples de las que estaban compuestos. Tras este análisis se llegó a la conclusión de que existen infinidad de estas estructuras, aunque es posible construir hipertextos narrativos de gran riqueza mediante el uso de cuatro tipos:

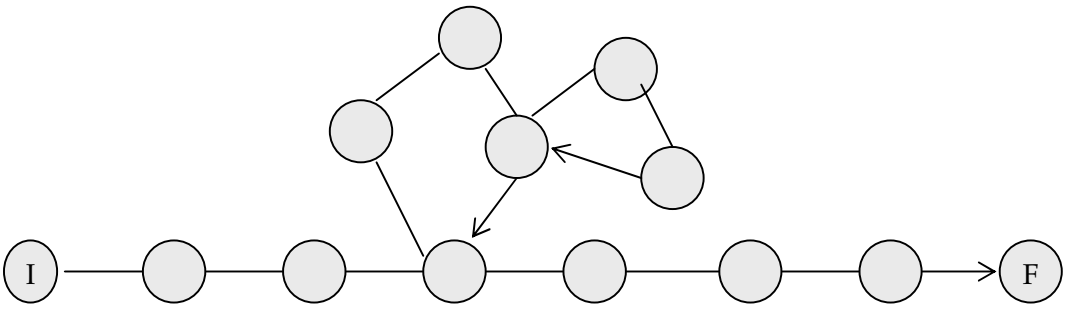
- **Estructura lineal**. Secuencia que representa el tipo de estructura de la literatura tradicional. Se utiliza para avanzar en el relato y es necesaria para evitar que el usuario se vea forzado a activar constantemente enlaces para realizar la lectura, lo que resultaría agotador. Es el tipo de estructura que conforma la secuencia principal de lectura en el hipertexto.



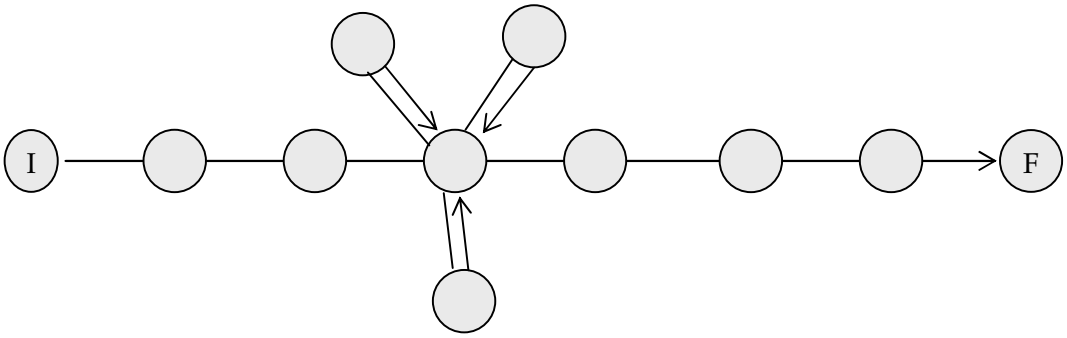
- **Secuencia alternativa**. Representa una ruptura en la linealidad del relato. Se trata de una secuencia lineal que tiene su origen en un nodo de la secuencia principal y su final en un nodo diferente de la misma, de modo que los nodos que la componen son excluyentes de los nodos de la secuencia principal situados entre origen y final de la secuencia alternativa. Se puede utilizar para contar experiencias que podrían ocurrir de forma paralela a otras, para narrar un hecho centrado en cómo lo vive un determinado personaje o para seguir en momento puntual a un personaje, lugar, hecho…. La razón para el uso de este tipo de estructura es que puede haber ocasiones en que el deseo de profundizar más en algo haga avanzar la historia al mismo tiempo.



- **Bucle abierto.** Representa también una ruptura en la linealidad del relato. Constituye un caso particular del tipo anterior. Se trata de una secuencia lineal cuyo origen y destino es un único nodo de la secuencia principal, de modo que cuando se ha pasado por los nodos que la componen, se vuelve al nodo del relato central del que se partió. Este tipo de estructura fue propuesto por de las Heras en [26], como el único tipo de estructura necesaria para crear un hipertexto bien diseñado. Son especialmente útiles para insertar digresiones, información adicional sobre un personaje concreto, etc. Implican, básicamente, una ampliación de lo narrado.



- **Secuencia tipo satélite.** Es la última de las secuencias que representan una ruptura en la linealidad del relato. En este caso se trata de una estructura formada por un nodo central del que parten diferentes ramas, formadas también cada una de ellas por un único nodo, y que tienen como origen y destino el nodo central. Aunque en principio este tipo de secuencia pudiera no parecer de especial relevancia, es en realidad útil hasta el punto que una de las primeras obras hipertextuales y de mayor importancia creada hasta el momento la usa en exclusiva en su estructura. Esta obra es "*Afternoon, a story"*(ver [30]), un clásico de la narrativa hipertextual. Este tipo de estructura es recomendable, por ejemplo, cuando se quiere presentar un mismo hecho desde el punto de vista de diferentes personajes.



Con la introducción de estas estructuras como los "bloques" con los que construir el hipertexto se ha dado el primer paso en lo concerniente al primero de los dos inconvenientes expuestos: la sensación que experimenta el lector inexperto al navegar por el hipertexto. Sin embargo, aún se puede llegar un poco más lejos a este respecto. De forma adicional, para estas estructuras se incluyen una serie de reglas de introducción que ayudan a este propósito:

- La herramienta no permite añadir ningún tipo de estructura hasta que la secuencia principal no ha sido introducida en el área de trabajo. Esto fuerza a que la estructura principal de la historia, la que lleva el peso del relato, sea una secuencia lineal. Este tipo de estructura permite la convergencia de los bucles hacia un único final, evitando de esta forma la posibilidad de varios finales. Esto implicaría permitir al lector influir sobre la historia, lo que podría provocar precisamente lo que se pretende evitar, una sensación de desorientación en el lector.

- Es necesario seleccionar un nodo de referencia de entre los ya insertados en el área de trabajo con anterioridad a la inserción de cualquier tipo de secuencia que no sea la lineal. Esta restricción obliga al grafo a mantener una estructura dentro de unos determinados límites, impidiendo que se entrecrucen los bucles. Esta posibilidad dispararía la complejidad del diagrama, provocando el riesgo de crear un caos en la estructura.

- Durante el proceso de inserción de cualquier tipo de estructura, tras hacer clic sobre el botón adecuado de la barra izquierda, la herramienta solicita al usuario la introducción del número de nodos que quiere que conformen la nueva secuencia. De nuevo con el fin de mantener la estructura hipertextual dentro de los límites de lo razonable y lo manejable, la herramienta limita el número máximo de nodos que se pueden introducir para cada tipo de secuencia. En el caso de la secuencia principal lineal, podrá estar formada por un máximo de cincuenta nodos. En el caso de cualquier otro tipo de estructura, el máximo está en veinte. Estos límites permiten formar una estructura lo suficientemente compleja como para que el autor no vea restringidas sus opciones de creación, y a la vez evitan que la red crezca de un modo ilimitado.

Tras la introducción de estas reglas se ha logrado aumentar el control sobre la forma y el tamaño de la red hipertextual, pretendiendo con ello facilitar de una forma natural al autor la construcción de hipertextos que eviten esta sensación de desorientación que se ha venido mencionando durante este apartado.

Lo único que queda por resolver, por lo tanto, es el hecho de que los lectores de las obras creadas con HyperAuthor puedan tener la sensación de estar perdiéndose algo al avanzar en la lectura y tomar decisiones en cuanto el camino a recorrer.

En este punto, se llegó a la conclusión de que por el momento sólo serían incluidas en la herramienta aquellas estructuras básicas que permitiesen al lector volver al punto en el que se comenzó una bifurcación. Dicho en otras palabras, sólo se podían permitir, además de obviamente la estructura lineal principal, aquellas estructuras que rompieran la linealidad del relato pero tuvieran como origen y final el mismo nodo. Esto obligaba a dejar fuera de la herramienta a las secuencias alternativas, por lo que de forma definitiva en la herramienta se incluyeron tres tipos de estructura: lineal, bucle abierto y secuencia tipo satélite.

A pesar de que la no utilización de las secuencias alternativas pueda parecer una pérdida demasiado grande como para ser asumida, no hay nada más lejos de la realidad. Los tipos de estructuras que terminan siempre en el nodo del que partieron presentan una serie de ventajas como son la suavidad y la homogeneidad, además de tener también su contribución en cuanto a evitar la sensación de desorientación o pérdida. La combinación de estas estructuras proporciona al autor la suficiente flexibilidad como para poder atender todas sus necesidades y además, desde el punto de vista del lector, la obra final presentará una continuidad y coherencia durante la narración que no es seguro que se alcanzase de otra manera.

Con esta decisión se consigue que el lector tenga la opción (que no la obligación) de leer y pasar por cada uno de los diferentes caminos de lectura que pueda ofrecer el hipertexto. De este modo se evita que la elección de un determinado enlace frente a otro situado en el mismo nodo conduzca a los usuarios hacia la sensación de "pérdida" de la parte de la historia ofrecida por ese otro enlace no seleccionado. Mientras que en estructuras como la formada por secuencias alternativas una elección implica un avance en

el relato de modo que no se vuelve a tener opción de navegar por los enlaces descartados, las estructuras incluidas en la herramienta ofrecen todo lo contrario: el usuario acabará siempre en el mismo punto en el que se comenzó la bifurcación escogida, de modo que si hubiera otras opciones de lectura diferentes en ese mismo nodo, al volver a él el usuario es libre de navegar por ellas si así lo desea.

Por otro lado, este tipo de estructuras ofrecen una ventaja adicional. No sólo evitan la desorientación que se puede experimentar durante la lectura de hipertexto y la sensación de estar perdiéndose cosas que sufre el lector al realizar una elección, sino que además ayudan a educar a los lectores en un nuevo tipo de literatura. Como ya ha sido expuesto y razonado, las particularidades que presentan este tipo de secuencias seleccionadas ayudan a que el hipertexto resultante no esté excesivamente alejado de aquello a lo que los lectores están acostumbrados, la página impresa. De esta manera les es más fácil asimilarlo y les prepara para un nuevo avance hacia la literatura hipertextual madura.

En nuestro caso, todo lo comentado en los párrafos anteriores se traduce en una serie de restricciones que debe respetar el grafo que representa la estructura hipertextual de las obras:

- Un eje tiene un solo origen y un solo destino.
- Los ejes en una estructura lineal y de inicio en un bucle abierto son bidireccionales.
- Los ejes que representan el fin de una bifurcación son unidireccionales.
- De un nodo pueden salir uno o varios ejes.
- Un nodo puede ser destino de uno o varios ejes.
- Cada nodo debe ser origen o destino de al menos un eje (no existen nodos sueltos, da lugar a secuencias lineales).
- Un nodo que es origen de una bifurcación debe ser también fin de la misma. (Da lugar a los bucles abiertos y las estructuras tipo satélite).

Los puntos anteriores definen el conjunto de restricciones formales que definen el modelo de grafo permitido para representar la estructura hipertextual. Nótese que el uso del adjetivo "formal" no es caprichoso, sino que obedece al hecho de que este conjunto de normas establecen limitaciones a la forma que debe tener el grafo con el que se trabaje.

En este conjunto de restricciones formales descansa el tratamiento a nivel estructural de las obras creadas, lo que en el apartado 4.2.2 se ha denominado "consistencia de la representación gráfica de la estructura". Llevando a cabo y comprobando las restricciones anteriores en el grafo, la interfaz gráfica garantiza que el diagrama correspondiente sea correcto en todo momento. La interfaz gráfica debe, por lo tanto, controlar y asegurar el cumplimiento de estas restricciones en los grafos que se dibujen en pantalla.

Para lograr este objetivo y tener la seguridad de que el grafo representado es siempre válido, se adoptaron una serie de medidas que se exponen a continuación. Es importante hacer notar que entre esta serie de medidas se encuentran también aquéllas

que ya han sido expuestas y cuya finalidad no era simplemente mantener la validez del grafo, sino además evitar los inconvenientes que presenta el hipertexto. Estas medidas ya fueron detalladas en párrafos anteriores, por lo que no aparecen en la siguiente lista.

- Durante la introducción de cada tipo de estructura, ya se ha comentado que la herramienta interroga al usuario sobre el número de nodos que desea que compongan la nueva secuencia. Sin embargo, el número de nodos que componen una secuencia puede ser incrementado o disminuido por parte del usuario en cualquier momento si así lo desea. Este proceso no debe comprometer la validez del diagrama, por lo que la interfaz debe realizar una fase de comprobación y procesamiento para cada una de estas funcionalidades para asegurar que:

  - El alargamiento de una secuencia se lleva a cabo mediante la introducción de nuevos nodos sueltos en la secuencia. Para ello, al igual que en el proceso de introducción de cualquier otro tipo de secuencia, se debe seleccionar previamente un nodo que servirá como referencia para la introducción del nuevo nodo, y decidir si se quiere introducir a derecha o izquierda del mismo. La interfaz se encarga de corregir los enlaces afectados por esta operación.

  - El acortamiento de las secuencias se puede realizar mediante el borrado de nodos sueltos. Durante el proceso de supresión la interfaz debe comprobar que el nodo seleccionado no sea origen o final de ninguna bifurcación antes de ser borrado; si lo es, las estructuras que "cuelgan" de él deben ser también suprimidas a la vez para evitar que esas estructuras queden sueltas. Si hay más de un nodo seleccionado para suprimir, antes del borrado la interfaz debe comprobar que son nodos consecutivos y forman parte de la misma estructura.

- También es responsabilidad de la interfaz gráfica el procesado necesario para garantizar la validez del diagrama durante los procesos de copiado, cortado y pegado. Los nodos seleccionados deben cumplir los mismos requisitos que los ya explicados en el punto anterior para los nodos que van a ser borrados: deben ser consecutivos y formar parte de la misma estructura. En cuanto al proceso de pegado, es similar al de inserción, se debe seleccionar un nodo que actuará como referencia. Los nodos copiados o cortados pueden insertarse a izquierda o derecha del nodo seleccionado como referencia. De forma posterior al pegado, la interfaz chequea y corrige todos los enlaces que se hayan visto afectados por esta operación.

Mediante el cumplimiento de todos los puntos anteriores, lo que implica capacidad de procesamiento en algunos casos y la restricción de las posibilidades ofrecidas al usuario en otros, la interfaz es capaz de asegurar en todo momento la consistencia del diagrama dibujado en el área de trabajo.

B. Acceso rápido a las estructuras.

En cuanto a la accesibilidad de las estructuras básicas en el programa, se decidió realizar un diseño actual y atractivo que favoreciera además la familiarización del autor con las mismas. Las estructuras se sitúan en un área auxiliar a al izquierda en una barra de herramientas especial para ellas, dejando libre la zona central para la construcción del diagrama. De esta forma las estructuras están siempre visibles y sólo se activan aquellas que pueden ser insertadas en ese momento, por lo que el autor tiene siempre visible las posibles opciones y aprende el uso de cada una de ellas.

Como en cualquier otra barra de herramientas, el mecanismo para insertar una estructura nueva en el área de trabajo consiste en hacer clic sobre la estructura deseada. Sin embargo, a este nivel existe una diferencia entre insertar la secuencia principal de la historia o cualquier otro tipo de estructura. La secuencial lineal principal sólo puede insertarse si el área de trabajo está vacía, mientras que las secuencias de ramificación pueden insertarse en cualquier momento. En este último caso, antes de hacer clic sobre la estructura que se desea insertar, se debe seleccionar el nodo que se quiere que actúe como inicio y fin de la bifurcación que va a implicar la inserción de la estructura.

Una vez se haya hecho clic en la estructura, tanto para la principal como para cualquier otra, aparece en pantalla un diálogo que interroga al usuario sobre el número de nodos que desea que contenga la nueva secuencia. El usuario selecciona el número que desea y tras pulsar aceptar, la nueva secuencia aparece en el área de trabajo ya conectada con el nodo que se eligió como origen. De este modo escoger una nueva secuencia e integrarla en el diagrama es algo inmediato.

C. Interfaz gráfica familiar.

Con el fin de que al usuario le resulte la aplicación lo más familiar posible, se escogió seguir un modelo similar al de un procesador de texto común, ya que éstas son las herramientas con las que más familiarizados se encuentran los autores de literatura. Al fin y al cabo, al igual que un procesador de texto (aunque con funcionalidades especiales), HyperAuthor pretende ser un área de trabajo en la que el usuario se sienta cómodo en la escritura de hipertexto, permitiendo de esta manera que su atención se pueda enfocar en la realización de tareas de índole más abstracta.

Por otro lado, esta semejanza que se pretende establecer, por asociación, reduce el rechazo inicial de los autores ante una aplicación que no conocen para abordar un género que no dominan. Además, es igualmente importante y necesario satisfacer las expectativas de los usuarios acostumbrados a encontrar determinadas facilidades en cualquier tipo de aplicación.

Se tomó, por tanto, la decisión de implementar en la interfaz gráfica, las siguientes utilidades:

- Apertura simultánea de varios grafos mediante la utilización de pestañas.
- Copiado/Cortado/Pegado de nodos o de estructuras dentro de un mismo grafo o de un grafo a otro.
- Funciones de deshacer/rehacer cambios recientes.

- Selección de múltiples nodos o de estructuras completas
- Menús con las funciones estándar más todas aquellas específicas de HyperAuthor.
- Barra de herramientas con redundancia de las operaciones más frecuentes del menú.
- Activación de teclas de acceso rápido
- Menú emergente accesible mediante el botón secundario del ratón con redundancia de las operaciones más frecuentes del menú.

Evidentemente, de este modo se mejora tanto la calidad del programa como la experiencia del usuario. Pero lo que es sin duda más importante, es que se consigue reducir la atención que un autor debe prestar a las tareas accesorias cuando usa la herramienta, consiguiendo, por lo tanto, aumentar la que dedica al diseño de la obra.

D. Representación gráfica de las estructuras.

Para la representación gráfica de cada tipo de estructura básica se decidió escoger una representación tradicional semejante a la que se usa en grafos o diagramas, con cajas representando los nodos y líneas representando los enlaces.

En principio, esta representación fue escogida ya que, a pesar de no ser una simbología estándar, es muy común encontrarla en todo tipo de diagramas o esquemas gracias a que ofrecen una interesante combinación de simplicidad y fuerza plástica.

Posteriormente, se decidió utilizar para la representación de cada una de las estructuras básicas una única figura que representara todas las posibles estructuras unidas. Para distinguir unas de otras se resaltó en cada una de ellas, usando un color diferente, la parte de la figura que se correspondiera con el tipo de estructura que se pretende insertar. De esta forma la propia representación de la figura completa da una idea de cómo se puede construir la totalidad de la estructura de una obra hipertextual a partir de estructuras básicas.

Por otro lado, las estructuras insertadas en la herramienta son representadas en pantalla de un color diferente dependiendo de la profundidad a la que se encuentren. Se entiende por profundidad el número de niveles que un nodo está alejado de la secuencia principal. De esta forma, una estructura básica que esté conectada directamente con la secuencia principal tendrá un color diferente a un mismo tipo de estructura conectada a un nodo que no pertenezca a la secuencia principal.

La introducción de colores en función de la profundidad de los nodos tiene dos efectos diferentes. Por un lado, la impresión del usuario es más favorable con herramientas que posean una interfaz vistosa y colorida, con una gran carga de diseño gráfico (sirva de ejemplo los propios sistemas operativos). Además, es necesario que la herramienta resulte atractiva y se evite provocar cualquier tipo de rechazo ya que los usuarios de la misma ya tienen una ardua tarea entre manos: aprender a escribir de una forma diferente, aprender a entender y crear un nuevo tipo de literatura.

En segundo lugar, se pensó que el hecho de seguir ciertas reglas en cuanto al color y a las formas podía ayudar a la comprensión de los conceptos subyacentes al hipertexto de forma que los autores los asimilaran más fácilmente. En esta línea se estableció el siguiente convenio:

- ❖ Para los menús y barras de botones:
  - Se utiliza el verde para representar los nodos en las estructuras a excepción de aquellos que simbolizan el tipo de estructura se quiere insertar, en cuyo caso aparecen coloreados en rojo.

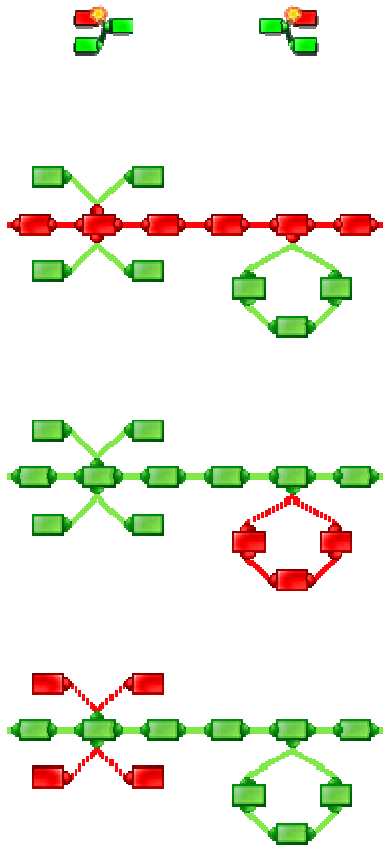


Figura 4.2 Posibles estructuras en HyperAuthor.

- ❖ Para la representación gráfica estructural:
  - Se asigna un color diferente a cada nivel de profundidad independientemente del tipo de estructura representada. Los colores son diferentes hasta un nivel de profundidad igual a 3, después comienzan a repetirse de forma periódica. Esto facilita la captación de la estructura global representada en pantalla.

- Los ejes en la dimensión estructural que aún no tienen su enlace equivalente en la dimensión de contenido, esto es, que aún no tienen ninguna palabra asignada en el texto del nodo, se representan en azul y se marcan con la palabra "<VACIO>".

- Los ejes en la dimensión estructural que ya sí tienen su enlace equivalente en la dimensión de contenido se representan de color negro y su nombre se corresponde con el nombre de la palabra en el texto del nodo con la que están relacionados.

- Los enlaces, vacíos o no, se representan por líneas sin ninguna terminación especial a excepción de aquellos que indican el inicio o final de una ruptura de una estructura lineal, en cuyo caso la línea está terminada en flecha indicando el sentido de la lectura.

De este modo se crea un "protocolo visual" mediante el cual la estructura hipertextual que se muestra en pantalla se puede analizar globalmente en un solo vistazo. Un ejemplo de la utilización de colores se muestra en la siguiente figura.

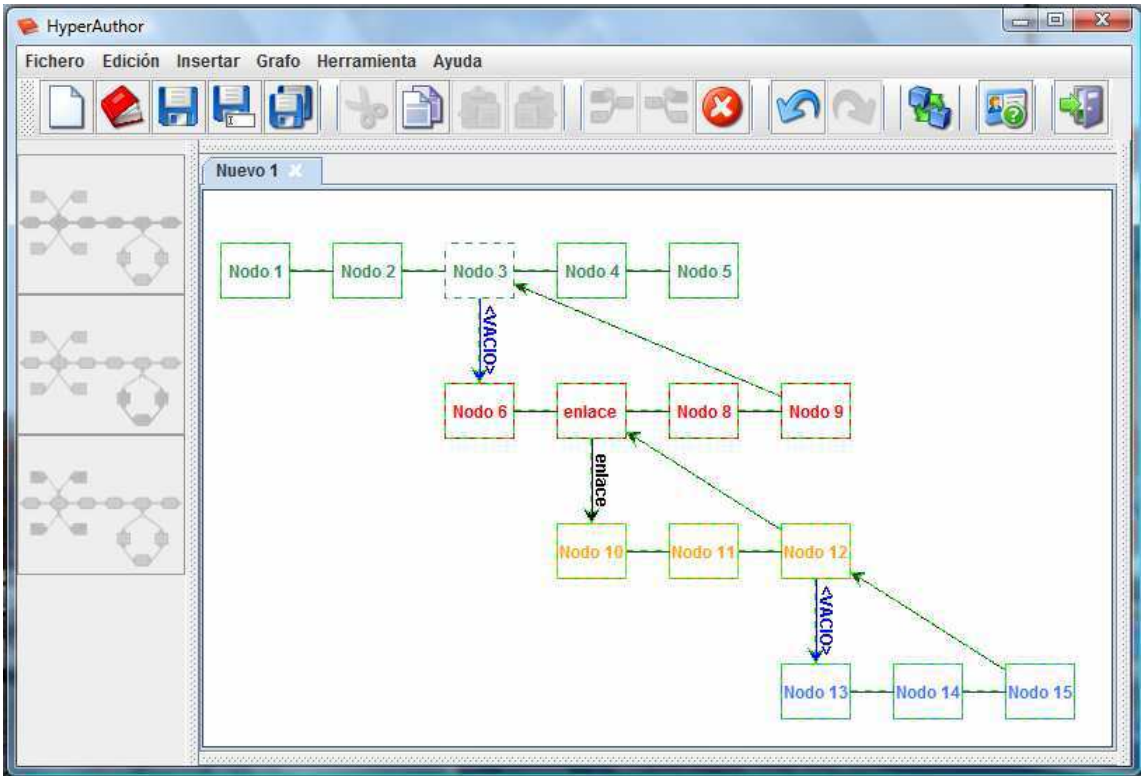


Figura 4.3 Esquema de colores empleado en HyperAuthor

Este convenio se muestra especialmente útil cuando se está trabajando con diagramas complejos, ya que los colores ayudan al autor a distinguir los distintos niveles de los que está formada la estructura completa mientras que las flechas de los enlaces ayudan a identificar el tipo de estructura representada. De esta forma se puede distinguir cada una de las estructuras tipo de las que está compuesta la estructura global a simple vista, lo que ayuda a realizar una composición mental de los posibles caminos de lectura de una forma rápida y sencilla.

E. Conexión de las dos dimensiones de diseño.

Como se ha comentado ya de forma reiterada, el proceso de diseño y creación de una obra hipertextual debe estar dividido en dos dimensiones que, al menos durante las primeras fases de escritura, pueden considerarse independientes: la dimensión estructural y la dimensión de contenido. Hasta ahora la mayor parte del esfuerzo y del proceso de diseño se ha centrado en lo que puede resultar más novedoso para el usuario final de la herramienta, es decir, la dimensión estructural. Sin embargo, la dimensión de contenido es algo que no se puede dejar olvidado.

Se ha planteado como uno de los requisitos fundamentales de la herramienta la conexión de estas dos dimensiones, de forma que, aunque sean en cierto modo independientes la una de la otra, ambas se puedan resolver en la propia herramienta.

La idea consiste fundamentalmente en que, una vez diseñada la estructura del hipertexto, desde la herramienta se facilite el acceso a la edición del contenido de cada uno de los nodos que forman parte de la misma. La manera más fácil e intuitiva de conseguirlo, a nuestro parecer, consiste en hacer aparecer en pantalla un editor de texto cada vez que se hace doble clic sobre uno de estos nodos.

Un editor de texto es un entorno con el que los autores de literatura tradicional, quienes serán los usuarios finales de la herramienta, ya están familiarizados. La elección de una interfaz similar a la de un editor de texto era, por lo tanto, una buena opción para cubrir el diseño de la dimensión de contenido. Por otro lado, este editor de texto debería incorporar algunas funcionalidades adicionales a las de la simple edición de texto necesarias para la tarea específica que se quería abordar.

Llegados a este punto, desde el punto de vista de la implementación había dos opciones posibles: desarrollar desde cero un editor específico completamente adaptado a las necesidades específicas de la herramienta, o utilizar un editor ya existente e integrarlo en la herramienta, añadiéndole las funcionalidades específicas de las que careciera.

Después de un análisis en el que se evaluó, para cada una de las dos posibilidades, la relación entre el esfuerzo y tiempo de implementación frente al resultado obtenido, se llegó a la conclusión de que la opción más viable era la de utilizar un editor ya existente que cubriera la mayor parte de la funcionalidad e introducir en él, durante la fase de integración, aquellas tareas específicas necesarias. Se llegó a esta conclusión no sólo porque un editor ya existente estaría dotado de una interfaz atractiva y de todas aquellas funcionalidades básicas a las que los autores de literatura ya están acostumbrados, sino

porque además esto permitiría dedicar la mayor parte del esfuerzo en la programación a aquellas funcionalidades específicas que aportarían riqueza a la solución global.

El editor seleccionado para ser integrado en la herramienta con este propósito es *SimplyHTML* (ver [34]). Este editor está escrito en Java y se distribuye junto con su código fuente completo así como con una documentación altamente detallada sobre su funcionalidad, uso y desarrollo, lo que sin duda facilitó la tarea de adaptación e integración.

SimplyHTML puede ser utilizado como una aplicación independiente o como un componente java destinado al procesamiento de texto enriquecido. Este editor almacena los ficheros que genera como documentos HTML en combinación con hojas de estilo (CSS). Este fue otro de los factores decisivos para su elección, ya que la utilización de tecnologías independientes de la plataforma como lo son Java, HTML y CSS tiene como consecuencia que el editor pueda ser utilizado en casi cualquier máquina.

Por todas estas razones, *SimplyHTML* se perfiló como el mejor candidato para cubrir los objetivos perseguidos por HyperAuthor. Se detalla a continuación el proceso de integración de este editor hasta llegar a ser el editor de nodos de la herramienta, poniendo especial interés en la descripción de las funcionalidades que se han denominado específicas.

Entre estas funcionalidades hay que destacar la que sin duda es la más importante de ellas: la creación y edición de enlaces. El editor ya permitía por sí mismo la creación de hiperlinks tal y como los conocemos en la actualidad, con destinos como páginas web, ficheros existentes en disco, etc. Al seleccionar una palabra y pulsar el botón correspondiente se abría un cuadro de diálogo que pedía al usuario la introducción del destino, el ancla del enlace, etc.

Sin embargo, los enlaces de la herramienta debían ser de otro tipo, debían enlazar diferentes porciones de texto dentro de una misma obra, y esta opción no estaba contemplada. Este cuadro de diálogo no servía para los objetivos de la herramienta, por lo que fue necesario adaptarlo. Por otro lado, esta funcionalidad debía conectar de algún modo los enlaces en el texto con los ejes correspondientes en el diagrama dibujado en la herramienta, es decir, era necesario conectar la dimensión estructural y la de contenido sin llegar a confundirlas, y ésta era una tarea que también había que abordar y resolver.

En la solución propuesta en una primera etapa, las dos dimensiones de diseño estaban íntimamente ligadas en este aspecto. Esta unión tenía su base en el hecho, fundamental en el hipertexto, de que todo enlace que implique una ruptura de la linealidad debe tener asociado una palabra en el interior del texto del nodo. Por este motivo, en un primer momento se pensó que cada vez que desde la herramienta se introdujera una estructura no lineal era necesario preguntar al autor sobre la palabra perteneciente al texto del nodo que quería que estuviera asociada a ese enlace.

Sin embargo, esto ponía al autor en una situación incómoda. Se veía forzado a asignar una palabra a un eje (generando la correspondiente palabra en el texto), cuando

cabía la posibilidad de que todavía no se hubiera planteado siquiera cuál sería el texto que habría dentro del nodo o, en el caso de que el nodo en cuestión ya contuviera texto, el autor podría no recordar qué palabra de entre las escritas era la adecuada para el enlace.

Ésta era una situación poco deseable, por lo que se pasó a una solución que proporcionaba un desacoplo algo mayor entre las dos dimensiones de diseño. Esta vez, en lugar de preguntar al autor por una palabra en concreto, se optó por otra posibilidad. Cuando desde la herramienta se insertaba una secuencia no lineal, tipo bucle abierto por ejemplo, se abría el editor mostrando el texto contenido en el nodo. Además, sobre la pantalla del editor, se mostraba un cuadro de diálogo que preguntaba sobre la palabra del texto a usar como enlace.

Esta situación tenía algunas ventajas respecto a la anterior. Por un lado, este diálogo permitía al autor tener acceso al texto del nodo en el que se debía crear el enlace y bastaba con seleccionar en el texto la palabra escogida como enlace para que ésta se reflejara en el diálogo o, en el caso de que se deseara incluir una palabra nueva en el texto, situar el cursor en el punto deseado y escribir la palabra en el cuadro de diálogo. Por otro lado, en el caso de que el nodo estuviera vacío, el autor tenia la posibilidad de escribir el texto en el nodo para posteriormente crear el enlace.

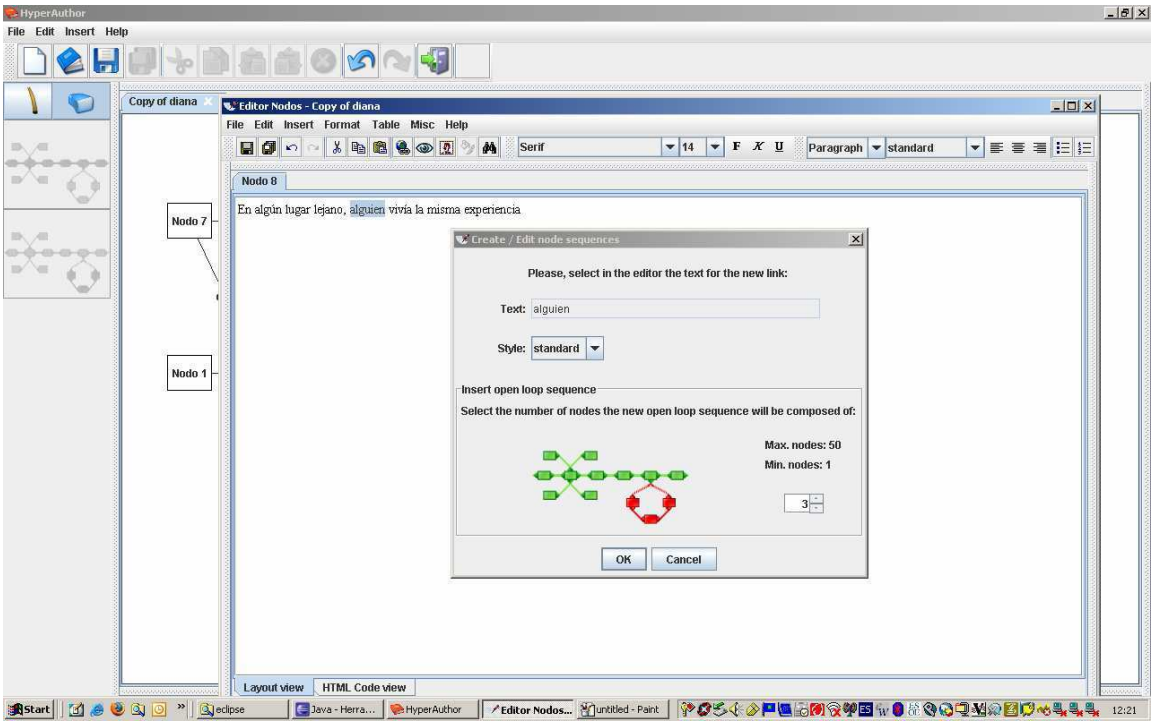


Figura 4.4 Diálogo de creación de enlaces empleado en versiones preliminares de HyperAuthor

A pesar de que ésta era una situación más conveniente que la anterior, tampoco fue la solución adoptada de forma definitiva ya que en ella se confundían las dos dimensiones de diseño: el autor podía verse forzado a introducir texto en un nodo sin tener previamente terminada la estructura de la red de hipertexto. Esto iba claramente en contra de los objetivos primeros de la herramienta, por lo que de nuevo hubo que "darle una vuelta" a este aspecto.

La solución definitivamente adoptada separa de forma casi total las dos dimensiones de diseño. Cuando en la herramienta se introduce una estructura no lineal, los ejes que implican una bifurcación se crean como vacíos, es decir, se crean sin tener una palabra asociada en el texto, y se marcan como tales con la palabra <VACÍO>. Estos ejes permanecen vacíos hasta que el autor pasa al diseño de la dimensión de contenido, ya que sólo se pueden asociar palabras desde el editor de nodos. Al editar un nodo del que surgen ejes que están vacíos (aún no tienen un enlace asociado en el texto del nodo), el editor de texto permite seleccionar una palabra para asociarla con cualquiera de los ejes vacíos del nodo, creando de esta manera un enlace.

Esto se lleva a cabo mediante un diálogo de edición de enlaces que permite establecer el texto del enlace, el nodo destino con el que se quiere que comunique el enlace que se va a crear y el formato del mismo. Cuando el enlace se crea en el texto, el correspondiente eje deja de estar vacío en el diagrama estructural. De la misma manera aparece ese mismo diálogo cuando lo que se selecciona es un enlace ya existente y lo que se quiere es editar sus propiedades.

Como ha quedado demostrado, la creación de enlaces en el texto de un nodo es una acción que pone en relación las dos dimensiones que hasta ahora se habían tratado de forma separada. Esta comunicación también se pone de relieve cuando se edita el texto o el título de un nodo, sin necesidad de haber creado necesariamente un enlace en él, lo que supone también la adaptación de esta funcionalidad previamente existente en el editor a las necesidades concretas de la herramienta. El texto de un nodo puede editarse sin más que escribir en el área de texto del editor. Para editar el título existe una opción en el menú que abre un cuadro de diálogo con el título actual y ofrece al usuario la posibilidad de modificarlo.

Cuando el título o el texto de un nodo se modifican, los cambios aparecen reflejados en la dimensión estructural de forma que el título por defecto que aparece dentro de cada nodo desaparece y aparece el nuevo título. Si no se ha dado ningún título pero si se ha modificado el texto, lo que aparecen son las primeras palabras del texto a modo de título. Esta funcionalidad ayuda al autor a distinguir unos nodos de otros y, como ha quedado patente, constituye una ventana de comunicación entre las dos dimensiones o planos de diseño.

Cuando un grafo alcanza una complejidad, y por lo tanto también un tamaño, dignos de mención, es posible que el título asignado a un nodo no pueda ser mostrado por completo en su interior porque no exista espacio disponible. Esto podía llevar al autor a situaciones equívocas en las que no pudiera distinguir dos nodos sin acceder a ellos. En ese momento se pensó que sería una buena idea que, al colocar el ratón sobre un nodo, su

título apareciese completo en la típica etiqueta amarilla que utilizan los *tooltip* de los menús o algo similar, por lo que esta funcionalidad extra quedó definitivamente implementada en la versión final de la herramienta.

Este mecanismo de edición remarca las diferencias entre una estructura, como el concepto abstracto que representa, y una secuencia particular en un hipertexto en particular, ya que es precisamente éste el momento en el que el autor particulariza los nodos para un propósito en concreto dentro de la obra. Hasta el momento las estructuras eran genéricas y neutras, y es mediante el establecimiento de su contenido que se convierte en concreta.

El editor permite no sólo la inclusión de texto, sino también de imágenes, con lo que cumple con otro de los requisitos de la herramienta. El requisito no hacía referencia exclusivamente a las imágenes, sino que hablaba de multimedia. Sin embargo, los elementos multimedia quedan fuera de la herramienta porque, aunque desde el punto de vista de la autoría puedan aportar riqueza a la obra, estos elementos fuerzan un ritmo interno en el espectador o lector y esto va en contra de los objetivos de la propia herramienta.

El multimedia puede reforzar lo transmitido, pero sólo si llega en el momento justo, y eso es habitualmente imposible de saber en un proceso de lectura. El video y similares se han descartado porque rompen el proceso de intimidad con el relato: pueden destruir y condicionar lo imaginado, además de romper el ritmo interno del lector. Lo mismo aplica a las imágenes demasiado explícitas.

Sin embargo, las imágenes sí se soportan en la herramienta porque el uso de ilustraciones de trazos difusos, en las que recaiga la fuerza en la rotundidad de las líneas o en el uso del color pueden resultar muy útiles. Se busca que las imágenes contribuyan a crear una atmósfera y no que describan lo narrado. Las imágenes empleadas no han de buscar describir, sino apoyar el sentimiento y las sensaciones que el texto pretende producir en el lector. Las imágenes de los periolibros son un buen ejemplo. Aunque la herramienta no es capaz de controlar esto, sino que queda en manos del autor, sí permite la modificación y tratamiento de las imágenes y se pensó que, a pesar de que la fuerza ha de residir siempre en lo escrito, en las palabras, la restricción del uso de imágenes mermaría de forma considerable las posibilidades de la obra final.

En el proceso de integración se consideró también la posibilidad de controlar de alguna forma la cantidad de texto contenida en cada nodo. Es innegable que la pantalla no es un buen dispositivo de lectura, por lo que incluir en un nodo una cantidad de texto considerablemente grande puede llevar al hastío al lector. Un texto demasiado largo presentado en pantalla provoca rechazo, y puede provocar que el usuario lector pierda la noción de dónde se encuentra con respecto al conjunto de la obra. Por este motivo se pensó en limitar el tamaño máximo del nodo.

Por otro lado, el autor debe también pensárselo muy bien antes de poner páginas con muy poco texto, ya que puede provocar un fenómeno conocido como fragmentación. El texto de cada página debe ser lo bastante largo como para que el lector tenga tiempo de

"conectar" con él, debe dar tiempo a sumergirse sin hastiarse. Esto implicaba tener que limitar también el tamaño mínimo del texto, lo que dejaba poco margen a la creatividad del autor, quien, por otro lado, puede querer aprovechar en un momento dado el efecto que una cantidad de texto pequeño puede producir. Se decidió, por lo tanto, no tomar ninguna medida de control en lo referente al tamaño del texto.

Otra de las adaptaciones necesarias fue la de eliminar el menú "Abrir nuevo" así como el menú "Abrir fichero" del editor. Obviamente, estas funcionalidades deben estar incluidas en cualquier editor que se precie, sin embargo, en la adaptación a la herramienta pierden su sentido ya que ésta sólo permite acceder al contenido de un nodo haciendo doble clic sobre él en el área de trabajo de la herramienta.

Esta forma de acceso a los nodos implica la adopción de una nueva decisión de diseño. Desde el inicio se había dado por supuesto que la visualización de cada nodo se iba a hacer en una ventana diferente. Sin embargo, usando una aplicación con una interfaz tan compleja como la del editor de nodos, esta idea perdía fuerza. El editor implementaba la utilización de diferentes pestañas para poder trabajar con diferentes documentos a la vez, y esto parecía una buena solución para poder tener abiertos varios nodos, por lo que se mantuvo esta configuración.

Ya se ha comentado con anterioridad que la interfaz de la herramienta permite también el uso de pestañas para trabajar con varios documentos a un mismo tiempo. La adopción del uso de pestañas también en el editor permitía además mantener agrupados todos los nodos abiertos que pertenecieran a un mismo grafo, ya que se decidió abrir un editor de nodos diferente para cada grafo en uso en la herramienta.

F. Ayudas a la navegación.

La implementación de un modo de navegación en el editor de nodos forma parte también de las funcionalidades que le fueron añadidas durante el proceso de integración. Este navegador cumple con la finalidad de ofrecerle al autor la posibilidad de experimentar dentro de la propia herramienta un proceso de lectura similar (aunque sin funcionalidades adicionales) al del lector de la obra.

Con este fin se crearon unos botones específicos en la barra de herramientas que permiten activar o desactivar este modo de navegación. Cuando un nodo está siendo editado, sus enlaces se encuentran inactivos, lo que quiere decir que el hecho de hacer clic sobre ellos no tiene ningún efecto. Cuando se entra en el modo de navegación, la edición del nodo queda inhabilitada pero se pueden seguir los enlaces que aparecen en pantalla mediante el modo habitual de hacer clic sobre ellos.

Cuando el usuario de la herramienta activa el modo de navegación y hace clic sobre uno de los enlaces, se abre en el editor una nueva pestaña que pasa a primer plano y que contiene el texto del nodo destino del enlace activado. Durante la navegación, el usuario puede activar cualquiera de los enlaces del nuevo texto para ir a su destino o moverse libremente por las pestañas abiertas en el editor. Este modo puede desactivarse

desde el botón destinado a tal efecto en la barra de herramientas y de nuevo la edición de los nodos volverá a ser posible.

En este punto de la fase de diseño se tiene que abordar uno de los aspectos más criticados de las herramientas que trabajan con hipertexto. Cuando un enlace se activa, se abre el texto del nodo destino en una pestaña nueva pero se pierde de vista el nodo origen. Sería muy conveniente poder tener a la vista el texto origen y destino de un enlace a un mismo tiempo, ya que el autor podría, de esta manera, comprobar la homogeneidad y continuidad de su relato.

Para posibilitar esta vista simultánea de origen y destino de un enlace, en el editor se incorporó una funcionalidad adicional disponible sólo si el usuario así lo selecciona en las opciones de configuración del mismo. Esta funcionalidad consiste en la introducción de un segundo panel de pestañas que aparece dentro del área central del editor cuando se entra en el modo navegación y se activa un enlace. De esta manera el origen del enlace activado queda visible en uno de los paneles y el destino en el otro, posibilitando no sólo la vista simultánea de ambos, sino también la edición en cada uno de los paneles. Es importante destacar que en cada momento sólo uno de los dos paneles estará activo, por lo que todas las acciones que se realicen utilizando la barra de herramientas o los menús serán llevadas a cabo exclusivamente en el nodo abierto en el panel que se encuentre activo en ese momento en concreto.

Una vez el segundo de los paneles, o panel auxiliar, se encuentra visible, las reglas de apertura de nodos son las siguientes:

- Cuando se intenta abrir un nodo desde la herramienta, se comprueba si ese nodo ya está abierto en cualquiera de los dos paneles.

  - Si lo está, la pestaña en la que esté abierto pasa a primer plano y el panel que la contiene pasa a ser el activo.

  - Si no lo está, el nodo se abre en el panel que se encontraba activo cuando se hizo doble clic en el nodo.

- Cuando se intenta abrir un nodo al activar un enlace con el modo de navegación activo, de nuevo se comprueba si ese nodo ya está abierto en cualquiera de los dos paneles.

  - Si lo está, la pestaña en la que esté abierto pasa a primer plano y el panel que la contiene pasa a ser el activo.

  - Si no lo está, el nodo se abre en el panel contrario a aquél en el que se produjo la activación del enlace, de manera que se ofrezca al autor la vista simultánea de origen y destino de un enlace.

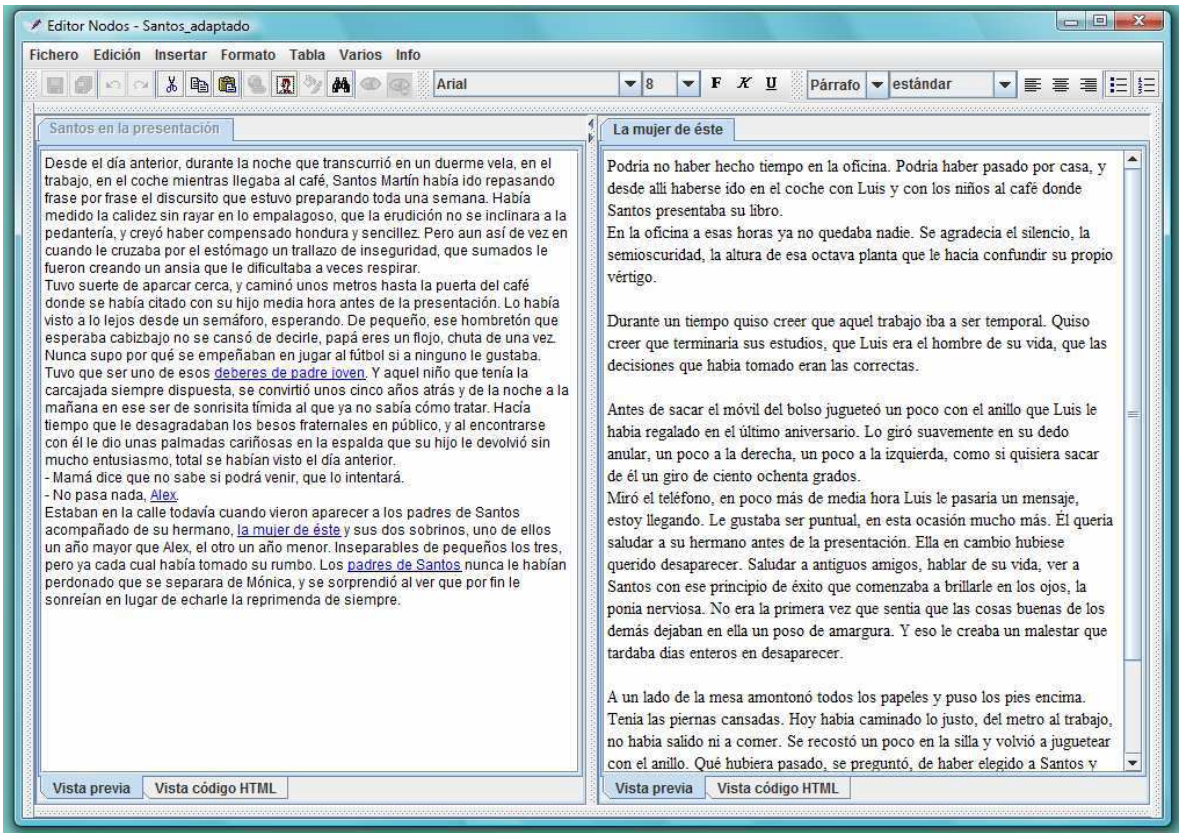


Figura 4.5 HyperAuthor en modo navegación cuando el uso del panel auxiliar de lectura está activo

Las pestañas dentro del panel auxiliar son tratadas de la misma forma que las abiertas en el otro panel. El autor puede editarlas, moverse por ellas o cerrarlas. Para cerrar el panel auxiliar o guardar el contenido de todas las pestañas que se encuentran abiertas dentro de él en un determinado momento basta con utilizar los menús destinados a tal efecto junto a los de cerrar o guardar convencionales.

Con la descripción de esta funcionalidad queda cerrado el proceso de integración del editor hasta convertirlo en el editor de nodos adaptado que finalmente queda incluido en la herramienta.

G. Personalización de la herramienta.

La inclusión del panel de pestañas auxiliar que se acaba de describir en el punto anterior como una funcionalidad que se activa por configuración (desde el menú opciones) forma parte de la personalización de la herramienta que puede llevar a cabo el usuario para trabajar con ella de un modo más cómodo.

Esta personalización de las herramientas es ventajosa para poder incluir opciones en la configuración de la misma que no implican un compromiso en el cumplimiento de la filosofía del modelo que hay detrás de ella ni en el logro de sus objetivos primeros pero

que sí ayudan a que el usuario trabaje en ella lo más cómodamente posible y sienta la herramienta como algo más suyo.

Además del mencionado panel auxiliar de pestañas, se ha introducido en la herramienta una segunda opción de configuración que afecta directamente a los ejes contenidos en el grafo en uso.

Por defecto, cada enlace que se crea en el área de trabajo de la herramienta mediante la introducción de estructuras que rompen la linealidad tiene asociada una etiqueta que muestra la palabra que actúa como enlace dentro del texto del nodo. Cuando un eje aún no tiene una palabra asociada, la etiqueta muestra la palabra <VACÍO>.

Sin embargo, cuando se está trabajando con diagramas que ya han alcanzado un cierto grado de complejidad, el espacio disponible dentro del área de trabajo de la herramienta puede parecer insuficiente. A este respecto, si hay algo prescindible dentro de la representación del grafo en pantalla es, precisamente, las etiquetas con el nombre de los enlaces asociados a cada eje, ya que esto forma parte de la dimensión de contenido y no de la dimensión estructural, con la que se está trabajando.

Por este motivo se añadió al menú opciones, accesible desde el menú herramienta, la posibilidad de trabajar en 3 modos diferentes dependiendo de la visibilidad o no de las etiquetas de los ejes. Estos 3 modos son:

- Modo visible: las etiquetas de los ejes del grafo son visibles en todo momento.

- Modo invisible: las etiquetas de los ejes del grafo permanecen ocultas en todo momento.

- Modo semi-visible: por defecto las etiquetas de los ejes del grafo aparecen ocultas y el usuario puede hacerlas visibles haciendo doble clic sobre el eje correspondiente. De la misma forma, en este modo, si una etiqueta se encuentra visible, se puede ocultar haciendo doble clic de nuevo en el eje correspondiente.

Tanto la funcionalidad referente al uso de un panel auxiliar de pestañas en el editor de nodos como la referente a la visibilidad de las etiquetas de los ejes de la herramienta se entienden como opciones de configuración de la herramienta y que, por lo tanto, afectan a todos los grafos y todos los nodos que se encuentren en uso. La activación o desactivación de estas funcionalidades permanecerá, por lo tanto, tras cerrar y volver a abrir la herramienta.

4.2.4 La lógica de negocio

Este bloque se compone del conjunto de funcionalidades que se realizan a nivel interno en la aplicación. En concreto, se debían satisfacer los siguientes objetivos:

- Implementar el tratamiento a nivel de contenido de los hipertextos, manteniéndolo en memoria y actualizando la interfaz de modo conveniente como respuesta a posibles cambios en el mismo.

- Ejecución de las tareas ordenadas por el usuario como respuesta a los eventos generados en la interfaz.

- Tareas auxiliares de la aplicación, entre las que se encuentran:
  - Persistencia de los datos
  - Gestión, a este nivel, del conjunto de obras hipertextuales que se encuentran en edición de manera simultánea.

Por supuesto, el punto fuerte del diseño de este bloque reside en lo concerniente al plano de contenido y la persistencia. Una vez establecido el modo en el que estas funciones se deben llevar a cabo, el resto de tareas auxiliares no requieren de demasiado esfuerzo de diseño.

Motivado por el hecho de que el editor de texto que se utiliza para la edición en el plano de contenido está basado en HTML y CSS, se llegó a la conclusión de que un buen modo de mantener la consistencia en la información perteneciente a este plano era almacenarla utilizando el lenguaje HTML. Por otro lado, el almacenamiento de la información en este formato no sólo era conveniente para la compatibilidad con el editor utilizado, sino que además esta forma de almacenamiento permitía al editor ofrecer una funcionalidad extra al usuario.

Hasta este momento se ha tenido en cuenta durante toda la fase de diseño el hecho de que la herramienta debía estar pensada y orientada a usuarios a los que no se les suponía un conocimiento técnico avanzado. Sin embargo, no hay que descartar la posibilidad de que algunos de los autores que accedan a la herramienta sí posean este conocimiento. Para este tipo de usuario está dirigida esta funcionalidad, y es que, basándose en el almacenamiento de la información de la dimensión de contenido en formato HTML, el editor de texto posee una pestaña de edición en la que se puede ver el código HTML correspondiente al contenido que se ha introducido en el área de texto. Los autores con el conocimiento necesario y que así lo deseen, pueden modificar este código HTML para "retocar" lo que se ha creado en la pestaña de vista previa.

Esta funcionalidad, que a primera vista puede parecer demasiado específica, se torna especialmente útil si tenemos en cuenta el hecho de que, por razones que se detallarán más adelante, el contenido de cada uno de los nodos no se almacena en disco en forma de un fichero HTML por cada nodo. Esto hace que, sin la existencia de esta pestaña de visualización de código HTML, se perdieran las siguientes ventajas que este lenguaje es capaz de ofrecer:

- El código HTML que representa el contenido de un nodo es legible por el usuario humano y fácilmente interpretable.

- Este formato es independiente de la aplicación, lo que supone una gran ventaja en caso de que se quiera rehacer o modificar.

En cuanto a la persistencia, por las ventajas expresadas con anterioridad, se decidió que una buena forma de guardar los programas en disco era mediante la creación de un archivo XML. De esta forma, los archivos generados podrían ser usados por otras aplicaciones con un coste mínimo en código de procesamiento. Por lo tanto, lo único que restaba por hacer era encontrar la forma de definir la estructura del hipertexto mediante *tags* XML, ya que una vez realizada esta tarea, para guardar la información perteneciente a la dimensión de contenido bastaba con incluir entre las propiedades de cada nodo definido en la estructura el código HTML que representa su contenido.

Para este propósito DOM era una alternativa tecnológica que facilitaba mucho la tarea. DOM (*Document Object Model)* es una librería java que permite la representación en forma de árbol de las obras hipertextuales con la que se trabaja en HyperAuthor. Esta representación permite el almacenamiento de las obras en memoria y ofrece, además, un amplio surtido de clases ya optimizadas que permiten operar con ellas.

El almacenamiento en disco utilizando DOM quedaba reducido únicamente a salvar la obra hipertextual representada como árbol DOM en un archivo XML. En resumen, la utilización de DOM en este bloque de la aplicación simplificaba considerablemente la implementación del mismo y proporcionaba ventajas en cuanto a posibles compatibilidades y separación del código.

La decisión estaba tomada, así que sólo restaba la construcción de una DTD *(Document Type Definition)* que definiese la estructura de una obra. Esta no fue una tarea demasiado complicada ya que las reglas impuestas por la herramienta en cuanto a la estructura del hipertexto simplificaron mucho la definición de la misma. Para la creación de la DTD, los puntos a identificar fueron los siguientes:

- Conjunto de elementos a incluir en la DTD y sus propiedades (atributos). En nuestro caso, cada nodo sería un elemento XML y tendría como atributos un identificador para distinguirlo del resto de nodos en la herramienta, su título, su contenido en formato HTML y el color (por motivos de representación en pantalla).

- Conjunto de elementos (hijos) que puede contener cada elemento: tipos de los hijos, cantidad de cada tipo y carácter de obligación si es necesario. En nuestro caso los enlaces de cada nodo aparecen como hijos de este y tienen como atributos el nombre y el destino. Además todo nodo debe contener al menos un hijo de tipo enlace, de forma que no sea posible tener en la representación en pantalla de la estructura nodos sueltos.

- Conjunto de valores de los atributos de cada tipo.

Si el lector tiene interés en los detalles de la definición de la DTD desarrollada para HyperAuthor, ésta se incluye en el Apéndice A de este documento.

Pues bien, aunque desde un principio se pensó que todo lo relativo a la persistencia quedaría con la definición de la DTD y la creación de un documento XML que guardara en disco tanto la información contenida en la dimensión estructural como la perteneciente a la dimensión de contenido, el uso de la herramienta puso de manifiesto que esto no era suficiente.

El editor de texto que se integró en la herramienta para ejercer como editor de nodos en la dimensión de contenido ofrecía, como ya se ha comentado, la posibilidad de incluir imágenes así como estilos personalizados para dar formato al texto. Cuando el usuario utiliza estas funcionalidades, el editor de nodos asociado al grafo en uso debe generar archivos adicionales al que ya se ha mencionado en formato XML para funcionar de manera correcta. Se genera, por lo tanto, una carpeta que contiene una copia de las imágenes que se han insertado en todos los nodos pertenecientes a ese grafo, y además, un archivo con la definición de la CSS que contiene todos los estilos que aparecen por defecto en el editor además de los definidos personalmente por el usuario.

La generación de más de un archivo por cada grafo que se ha guardado en disco no parecía una solución del todo limpia, que además podía provocar colisiones entre los archivos generados para dos grafos distintos ya que la carpeta de imágenes y el archivo que contiene la CSS usan siempre el mismo nombre, con independencia del nombre del grafo que se quiere guardar. Ante esta situación, pareció conveniente plantear una solución alternativa.

Dadas las ventajas de la utilización de un archivo XML en cuanto a legibilidad y compatibilidad, se creyó conveniente el mantenimiento de este tipo de archivo para almacenar la información relativa a estructura y contenido de la obra. Sin embargo, había que encontrar una manera de asociar los archivos auxiliares generados por el editor de nodos con el archivo XML generado por la propia herramienta. Esto se consiguió usando algo tan simple y usual como es crear una carpeta para cada obra que contiene todos los archivos asociados a la misma y comprimida usando el formato zip.

El uso de este formato de compresión, soportado de forma cómoda por Java, no restaba compatibilidad a la solución, ya que zip es un formato estándar y todos los archivos que contiene esta carpeta están escritos también en lenguaje estándar (como es XML y CSS). De esta forma, los archivos generados por la herramienta siguen siendo compatibles y aptos para el uso por parte de otras aplicaciones, sin más que incrementar de una forma mínima el coste en código de procesamiento para ejecutar la descompresión de la carpeta, lo que proporciona acceso inmediato a los archivos que contiene.

De esta forma, cuando la herramienta abre un archivo nuevo desde disco, lo primero que hace es descomprimir el archivo y llevar los archivos que contiene a un directorio temporal creado para cada una de las obras en uso en la herramienta. De la

misma manera, cuando se comienza un hipertexto desde cero y se le van añadiendo imágenes o estilos personalizados, los archivos creados por el editor de nodos se almacenan en ese directorio temporal. Cuando el usuario pulsa la opción de guardar, estos ficheros se copian a la carpeta creada en la ruta y con el nombre proporcionados por el usuario y se comprime. Los archivos temporales de cada hipertexto se borran al cerrarlo, y se hace una limpieza general de todos cuando se cierra la aplicación, de modo que el sistema de archivos del sistema operativo quede limpio tras su uso.

Pues bien, con esta elección se cierra la descripción y justificación del que ha sido finalmente el diseño de HyperAuthor. Sin embargo, se ha estimado oportuno dedicar un pequeño espacio de este apartado al diseño de la herramienta para destacar un hecho relevante y que quizá haya pasado desapercibido hasta el momento.

4.2.5 Necesidad de una doble estructura

Esta peculiaridad en la implementación de la herramienta surge de la etapa de la fase de diseño en la que se estudiaba las distintas alternativas para la implementación de los objetos gráficos con los que se iba a construir la estructura hipertextual. Como se ha explicado con anterioridad, era necesario que además de ser meros objetos gráficos (componentes), los nodos y ejes que formaban la estructura de la obra dispusiesen de la inteligencia necesaria para garantizar la conformidad y validez de la misma.

Las alternativas a este respecto eran básicamente dos:

- Construir una jerarquía de objetos que heredase de la clase Component de Swing (ver [31]) y proporcionarles la funcionalidad necesaria para implementar el grafo que se necesitaba. Es decir, generar un conjunto de componentes personalizados (ver [32]).

- Utilizar la librería genérica para grafos JGraph [33], adaptándonos a las capacidades gráficas que ésta ofrecía.

La opción más inmediata era, por supuesto, utilizar la librería JGraph, que contiene un numeroso conjunto de clases profundamente testeadas y suficientemente genéricas para el manejo de grafos (recordemos que, en la fase de diseño inicial, se decidió que en la interfaz gráfica se emplearían grafos para representar hipertextos, ya que ésta era la solución más natural).

Mientras, por las razones ya expuestas (ver 4.2.4), la lógica de negocio iba a utilizar DOM en lo referente a la persistencia y, por lo tanto, los hipertextos en este ámbito tendrían forma de árbol.

Y es ésta la peculiaridad que cabe resaltar sobre el diseño de la herramienta. Hasta cierto punto, la solución adoptada suponía duplicar la información. No estrictamente hablando, por supuesto, puesto que cada una de las estructuras que representa una obra

hipertextual contendría información de distinta naturaleza. Sin embargo, sí implica mantener dos estructuras de datos en paralelo.

A primera vista, puede parecer que mantener dos estructuras de datos "en paralelo" resta elegancia a la solución arquitectónica propuesta o que supone una mayor carga de procesamiento. Sin embargo, esto no es así. Ciertamente, el mantenimiento de la estructura por partida doble resta eficiencia en términos de memoria, pero la estructura de árbol que maneja DOM no es algo que tenga que actualizarse y validarse de la misma forma que la utilizada por JGraph, ya que sólo se emplea en temas de persistencia.

Esto quiere decir que la estructura en forma de árbol se crea en el momento de cargar una obra desde disco o al guardar una obra ya creada. En el caso de una operación de guardado, se recopila toda la información sobre estructura mantenida por la interfaz gráfica y toda la información sobre contenido mantenida por la lógica de negocio y se crea el árbol DOM. Para el proceso de apertura el proceso es el inverso, se opera sobre el árbol DOM y se extrae de él la información perteneciente a cada una de las dos dimensiones de diseño diferentes, pasándosela al bloque de la aplicación correspondiente.

Por lo tanto, el hecho de que esta estructura de árbol de DOM se use sólo en momentos puntuales de la operación con la herramienta, lejos de restar eficiencia a la solución arquitectónica, la convierte en una solución más robusta, ya que en el proceso de traducción de una estructura a otra se realiza un proceso de validación de la información que asegura la consistencia de la misma.

Se decidió, por lo tanto, que no era necesario buscar una solución alternativa y se mantiene la duplicación de estructuras utilizando las librerías disponibles para cada una: JGraph y JDom.

4.3 HyperViewer

Hasta aquí se han descrito con detalle las decisiones de diseño adoptadas en lo referente a la herramienta creadora, HyperAuthor, así como los motivos que influyeron en la toma de dichas decisiones. Por lo tanto, en lo que a la fase de diseño se refiere, resta sólo hacer lo propio con la herramienta de visualización: HyperViewer.

4.3.1 Requisitos

Llegado a este punto del documento el lector tiene ya una idea clara de los objetivos de la herramienta. Como apoyo a la herramienta creadora, cuyo diseño se ha desarrollado en el apartado anterior, se decidió crear una herramienta de visualización para el contenido creado que contara con características específicas que apoyara los principios del modelo en el que se fundamenta la herramienta.

Tal y como se ha explicado de forma reiterada en este documento, el pilar básico de las herramientas desarrolladas consiste en favorecer la creación de literatura hipertextual que, manteniendo los logros de la literatura tradicional, de un paso adelante hacia la creación de hiperficciones de calidad.

Sin embargo, dadas las características de la hiperficción, una hipertexto bien diseñado no es suficiente para evitar completamente la sensación de pérdida tan habitual entre los lectores cuando navegan por una red hipertextual. Ésta sensación, que como se explica en 4.2.3 A., es un de los principales inconvenientes que presenta el hipertexto, debe ser tratada no sólo en la dimensión estructural de la obra, sino también en la dimensión expositiva. Por lo tanto, se hace necesario el uso de un dispositivo lector que dote al usuario de herramientas de navegación que le ayuden en el proceso de lectura.

Por otro lado, es conocido que la lectura en pantalla no proporciona al lector una intimidad con el relato similar a la que le ofrecía la página impresa. Otro de los objetivos de HyperViewer debe ser, por lo tanto, conseguir la "adaptación" del concepto de página dentro del ámbito hipertextual sin eliminar por completo sus implicaciones tradicionales con el fin de facilitar esa unión entre relato y lector.

A modo de resumen se puede decir que la finalidad que se persigue es, por tanto, proporcionar a los lectores una herramienta que les facilite la lectura de hiperficción y con la que puedan "aprender" las peculiaridades de este nuevo tipo de lectura. De esta forma los lectores se familiarizarán con el nuevo género y será posible dar un nuevo paso adelante hacia la literatura hipertextual madura y de calidad.

Con estas premisas, y teniendo en cuenta las consideraciones expuestas en capítulos anteriores, la aplicación parte de los siguientes requisitos:

- Proporcionar un entorno de lectura en el que se juegue con la estructura interna de la página y el propio hecho de pasar página para semejar el nodo presentado en pantalla con la página impresa.

- Ofrecer al lector mecanismos y ayudas a la navegación, como puede ser un histórico de navegación que permita conocer el camino de lectura realizado por el lector y volver a un punto concreto del mismo en cualquier momento

- Mantener un enfoque claramente atrayente, simple e intuitivo. Eliminar, dentro de lo posible, barras de menús y de herramientas de forma que la mayor parte posible de pantalla esté destinada al área de lectura

4.3.2 Planteamiento general

A semejanza del diseño llevado a cabo con HyperAuthor, en HyperViewer se decidió utilizar también el patrón Modelo-Vista-Controlador (MVC) [28]. El patrón MVC es aconsejable en toda aquella herramienta que cuenta con una interfaz de usuario y, por otro lado, de esta forma el diseño de las dos herramientas se podía llevar a cabo de una forma similar.

De nuevo, como ya ocurría con la herramienta creadora, a la hora de llevar a cabo la implementación real de la herramienta nos encontramos con una pequeña dificultad, y es que el uso de Swing [32] hace que el desacoplo completo de los módulos vista y controlador sea complicado de lograr. Este es el motivo por el que en HyperViewer, y una vez más de un modo similar a como ocurría en HyperAuthor, la parte del procesamiento de eventos queda incluida en clases pertenecientes al módulo de la vista, de modo que el controlador queda semioculto dentro de la misma.

Por lo tanto, la particularización del modelo MVC para el caso de HyperViewer queda de la siguiente manera:

- Vista: Compuesto esencialmente por la interfaz gráfica que, como es bien sabido, es la encargada de interactuar con el usuario. En nuestro caso, se trata de una capa intermedia entre lector y relato que aporta características específicas necesarias en la dimensión expositiva para poder obtener una lectura profunda.

- Controlador: Es el eslabón que une la vista y el modelo. Se encarga de recibir los eventos de entrada así como de gestionarlos. La respuesta a estos eventos pueden suponer peticiones al modelo o a la vista. En nuestro caso, la gestión de eventos está integrada en la vista.

- Modelo: Se encarga del acceso a los datos y de la implementación de toda la funcionalidad de la herramienta. Por simplicidad, se decidió que este módulo se encargase también del almacenamiento de los documentos generados en discos (persistencia) y su recuperación.

Teniendo en consideración todo lo mencionado hasta el momento, la aplicación queda, a grandes rasgos, divida en dos grandes bloques:

- Interfaz gráfica: el conjunto de funcionalidades de índole gráfica y aporte de requerimientos específicos necesarios para el logro de una lectura profunda. También se incluye en este bloque parte de la gestión de eventos relacionados con la interfaz.

- Lógica de negocio: incluye todos los procesos y funcionalidades internas de la aplicación.

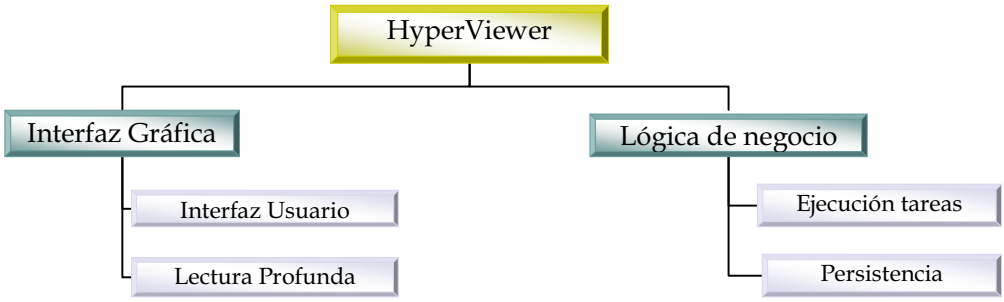


Figura 4.6 Arquitectura de HyperViewer.

4.3.3 La interfaz gráfica

De acuerdo con los objetivos asumidos en el desarrollo de esta aplicación, y dado el enfoque mediante el cual se pretende conseguirlos (descritos ambos en los capítulos anteriores) la interfaz de usuario de HyperViewer resulta ser una pieza muy importante en el desarrollo de este proyecto.

En la actualidad, la interfaz gráfica de cualquier aplicación o herramienta juega un papel muy importante, y en ocasiones determinante, en la opinión que el usuario se forma sobre la misma. Además, una interfaz bien diseñada puede facilitar al usuario final el uso de la herramienta, de forma que la interfaz no sólo debe resultar atrayente sino también intuitiva para el usuario. En HyperViewer, la interfaz debe cumplir además una serie de objetivos adicionales que permitan que el lector de hipertexto logre una lectura profunda similar a la obtenida con la lectura sobre página impresa.

El primero de los requisitos a tener en cuenta es aquel, ya mencionado con anterioridad, que hace referencia a la necesidad de la adaptación del concepto de página sin perder todas aquellas implicaciones positivas que su uso comporta. La pantalla del ordenador es un soporte al que los lectores no están acostumbrados y que actúa, de alguna manera, como una barrera entre el lector y el relato. Esto impide que el lector logre la intimidad con el relato, entendida como el autoconocimiento y el tiempo profundo necesarios durante el proceso de lectura.

A este respecto ya se introdujeron requisitos en la herramienta creadora que afectaban a la dimensión estructural de la obra, dejando en manos del autor consideraciones como la cantidad de texto incluido en un nodo o la cantidad de enlaces contenidos en ese texto. Una cantidad demasiado pequeña o grande de texto puede romper la conexión del lector con el texto, del mismo modo que una interacción excesivamente frecuente con la obra. Sin embargo, a pesar de que todos estos principios

hayan sido tenidos en consideración por el autor, es necesario lograr además un escenario de lectura adecuado que no rompa el ritmo interno de la lectura.

De las Heras (ver [26]) propone una forma de navegación que, dadas las características de la estructura de los hipertextos que la herramienta creadora permite construir, parecía adecuada para los propósitos de HyperViewer. La propuesta de de las Heras consiste en la división lógica (que no física, ya que esta división no es perceptible por el usuario) del área de lectura en tres partes aproximadamente iguales. De esta forma, al hacer clic sobre la franja izquierda de la pantalla, el lector vuelve al nodo anterior al que estaba visualizando y, de una manera similar, al hacer clic sobre la franja derecha avanza en el relato haciendo aparecer el nodo siguiente en una secuencia lineal.

Los hipertextos generados mediante el uso de HyperAuthor plantean un escenario en el que las secuencias básicas que componen la estructura total del hipertexto son fundamentalmente secuencias lineales. En primer lugar, es obligatoria la existencia de la secuencia lineal principal, que contiene el inicio y final de la historia. Por otro lado están los bucles abiertos y las estructuras tipo satélite que, si bien implican una ruptura de la linealidad en la secuencia de la que provienen, son en sí mismos secuencias lineales con origen y final en un mismo nodo, y el usuario se mueve dentro de ellas como si de una secuencia lineal al uso se tratara. En este escenario en el que la navegación de tipo anterior-siguiente es fundamental, una forma de navegación como la planteada por de las Heras es muy recomendable.

La ventaja fundamental que este tipo de navegación aporta consiste en la metáfora del paso de página. Desde el punto de vista del usuario, que se encuentra sumergido en un proceso de lectura, el hacer clic sobre la zona derecha de la pantalla para avanzar en el relato o sobre la izquierda para volver a lo leído es mucho más cómodo y natural que el hacer clic en un determinado botón. El hecho de tener que desviar la atención del relato para buscar el punto concreto en el que se encuentra el botón de “Siguiente” o “Anterior” puede hacer que se rompa ese ritmo interno de lectura tan anhelado y del que tanto se ha hablado ya.

Sin embargo, el movimiento con el ratón hacia la zona derecha o izquierda de la pantalla para hacer clic sobre ellas requiere de un esfuerzo mínimo y puede asemejarse, de una manera inconsciente y natural, con el hecho de pasar página en un libro tradicional. Este gesto es ya familiar para los lectores y en ningún caso representa un compromiso para la consecución del tiempo profundo.

Con esta consideración de diseño se resuelve el hecho de que un hipertexto debe estar formado por varias páginas (o nodos) como cualquier libro tradicional, y que el paso de una página a otra debe poder llevarse a cabo de una forma natural. Pero las ventajas de este tipo de navegación no terminan aquí…

La inclusión de las franjas anterior y siguiente en la pantalla permite distinguir entre dos tipos de navegación diferentes que pueden tener lugar durante la lectura del hipertexto. Cuando el usuario se mueve utilizando estas franjas, sabe que está navegando por una secuencia lineal, el relato avanza hacia su final del modo acostumbrado en la

literatura tradicional. Sin embargo, cuando el usuario desea tomar una nueva ruta, una que se salga de la linealidad de la secuencia en la que se encuentra para poder seguir la pista de un concepto, una idea, un personaje, un lugar o una historia que le interese en ese momento, entonces debe utilizar la activación de los enlaces que aparecen en el texto. Al pulsar sobre una de las palabras marcadas como enlaces, el visor abre y presenta en pantalla el contenido del nodo destino del enlace; para el lector queda abierto, en ese momento, un nuevo camino de lectura.

Estos dos tipos diferentes y diferenciados de navegación ayudan al lector a "orientarse" dentro del hipertexto, a saber con mayor exactitud su posición dentro de la red hipertextual y sobre todo, a hacerse una idea sobre qué es lo que viene a continuación de manera que el hecho de tomar una bifurcación en el relato no resulte desconcertante.

Hasta ahora se ha hablado de la funcionalidad de las zonas izquierda y derecha de las tres en las que el área de lectura de HyperViewer ha quedado divida. Resta, por lo tanto, explicar la funcionalidad asociada a la zona central de la pantalla. Esto conduce a este documento hasta la segunda de las consideraciones importantes de diseño, la que está íntimamente ligada con los enlaces.

En HyperViewer, al lector no le está permitida la toma de decisiones de navegación hasta que no ha completado la lectura del nodo. Poder saltar y tomar un nuevo camino de lectura en cualquier momento constituye un grado de libertad que compromete la creación de la obra. El texto del nodo junto con los enlaces y las imágenes que contiene conforman una única unidad de significado. Por lo tanto, para que el lector tenga la oportunidad de descubrir algo de sí mismo atendiendo a lo que el autor diseñó, esta unidad de significado debe ser asimilada de forma completa por el lector antes de que la herramienta le permita seguir los enlaces que contiene el nodo.

Con esta finalidad, cuando se accede a un determinado nodo, la herramienta no revela desde el primer momento qué palabras de las contenidas en el texto del nodo son realmente enlaces. Éstos aparecen imbricados en el texto pero no implícitamente marcados, por lo que se consigue que todas las palabras tengan el mismo valor para el lector cuando éste las lee.

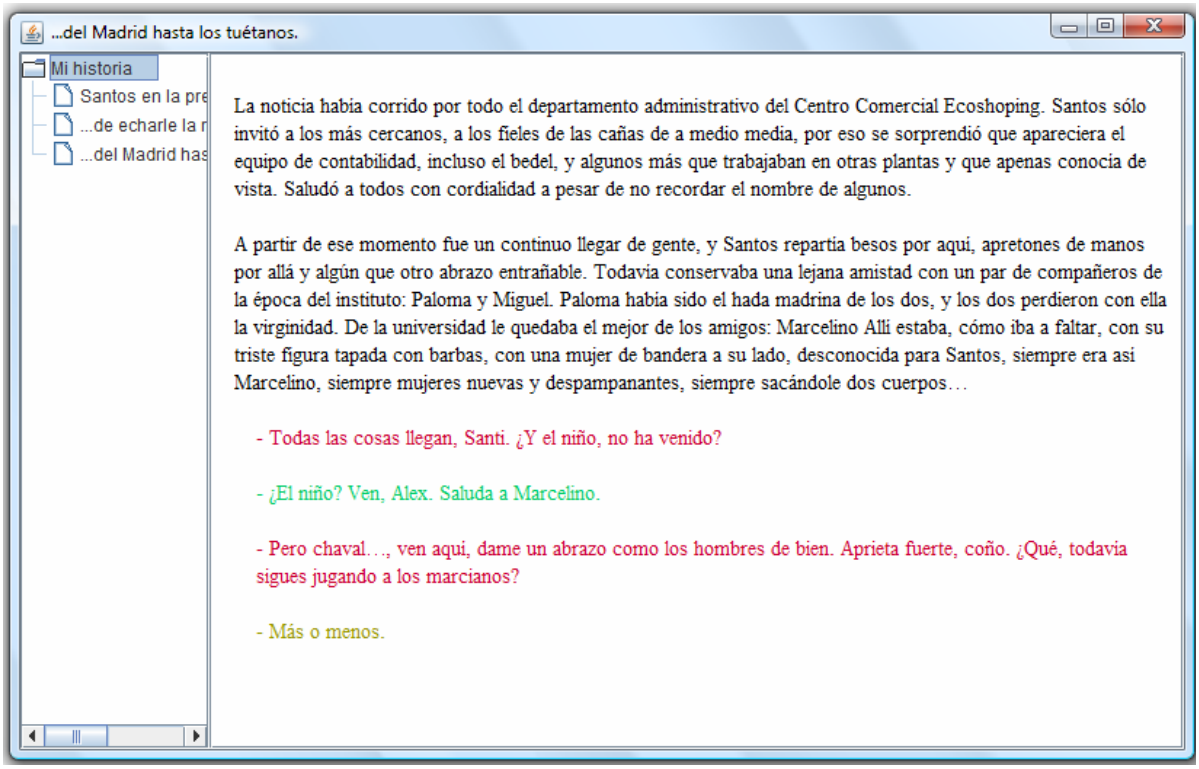


Figura 4.7 Vista de un nodo cuando se accede a él en HyperViewer.

Una vez terminada la lectura del nodo y asimilada la unidad de significado que constituye, mediante una pulsación en la parte central de la pantalla la herramienta hará aparecer los enlaces contenidos en el nodo. En este instante, el lector podrá decidir cuál de los caminos desea seguir. Con esto se logra un compromiso importante entre saciar el impulso interactivo del lector de hipertexto y la posibilidad de dejarle navegar demasiado pronto. Del mismo modo, si los enlaces están visibles, un clic sobre la zona central de la pantalla hará que estos desaparezcan.

Poner las decisiones de navegación al final no corta el proceso de lectura y, por otro lado, añade una pausa en el mismo que favorece la "digestión de lo leído", por lo que esta decisión quedó definitivamente implantada en la herramienta.

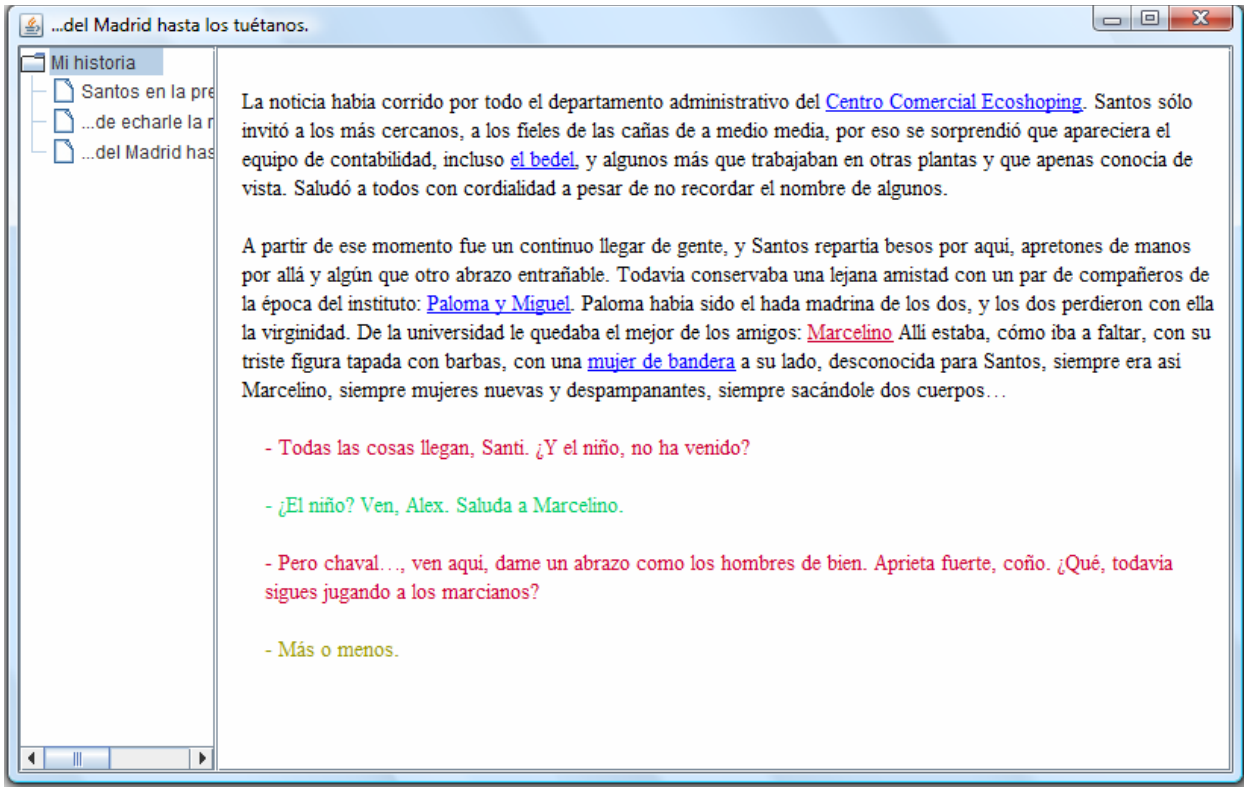


Figura 4.8 Vista de un nodo tras activar los enlaces en HyperViewer.

Sin embargo esta opción que ofrece la franja central del área de lectura puede llevar al lector a una situación confusa si se encuentra ante un nodo que no contiene enlaces. En este caso, el hecho de hacer clic sobre la zona central de la pantalla no obtendría ninguna respuesta por parte de la herramienta, ningún enlace podría aparecer resaltado ya que no existe ninguno. Pero esta es una información que el usuario no conoce. El lector no puede distinguir entre esta situación, perfectamente normal y habitual, y un fallo en la herramienta que provocara que, simplemente, los enlaces no se resaltaran en pantalla.

Para evitar esta posible confusión y con la convicción de que toda acción del usuario debe poder devolverle una respuesta por parte de la herramienta, se decidió adoptar una situación de compromiso. Cuando el usuario hace clic en la zona central del área de lectura pero no existen enlaces que resaltar en el texto, lo que se hace es resaltarle al usuario sus posibles opciones; esto es, aparecen resaltadas, aunque de un modo diferente al de los enlaces, las zonas anterior y siguiente (izquierda y derecha) de la pantalla. De esta forma el usuario siempre recibe una respuesta por parte de la herramienta cuando hace clic en la pantalla, y además, "informa" al lector sobre las alternativas que se le ofrecen en ese punto de la lectura.

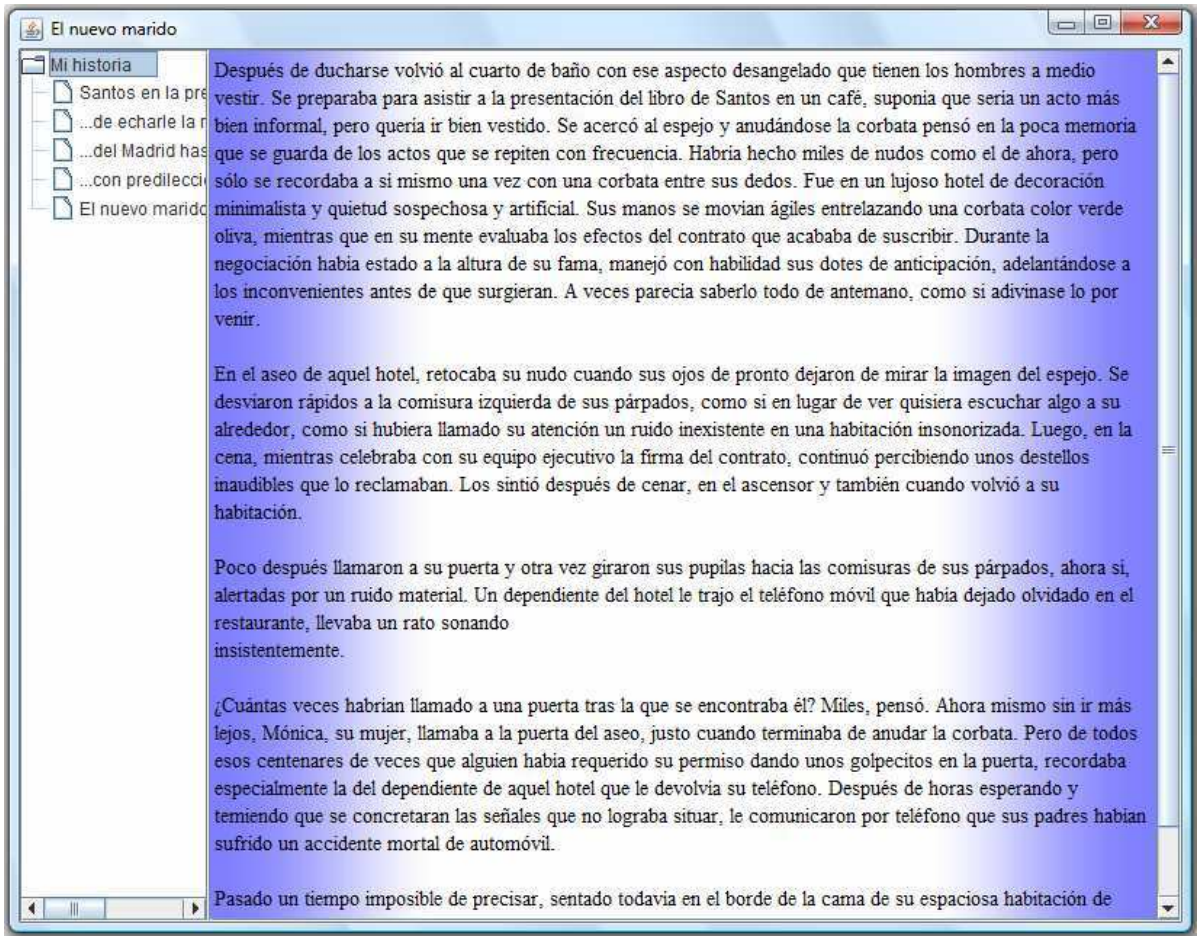


Figura 4.9 Vista de un nodo sin enlaces al tratar de activarlos en HyperViewer.

Hasta aquí se han expuesto las consideraciones de diseño relacionadas con aquellas necesidades específicas de un dispositivo lector para hipertexto que pretende favorecer el tiempo profundo entre sus lectores. Con anterioridad se expuso también la conveniencia de la implementación de mecanismos de navegación que ayudaran al lector a no sentirse perdido dentro de la red hipertextual. De estas ayudas a la navegación se va a tratar en lo sucesivo.

Uno de los mecanismos de navegación básicos que debe ofrecer una herramienta de visualización como HyperViewer es el acceso rápido a cualquiera de los nodos ya visitados que incluya la posibilidad de la vuelta inmediata al nodo actual. Con este tipo de ayuda el lector siente que, a pesar de haberse adentrado en un mar desconocido representado por la red hipertextual, no ha perdido totalmente de vista la orilla, representada por aquellos nodos que ya han sido visitados y que de alguna forma aportan seguridad ante la incertidumbre de lo que aún queda por descubrir. Hasta que los lectores se familiaricen con este nuevo tipo de literatura, este tipo de sistemas representan una ayuda necesaria para ellos.

Por otro lado, parece interesante poder unir esto con un histórico de los nodos ya visitados y que representen "cómo ese lector leyó la historia" (ya que un mismo hipertexto puede ser leído de diferentes maneras). En nuestra herramienta, este histórico aparece visible en el lado izquierdo de la pantalla en forma de árbol, de manera que cada vez que el usuario visita un nodo, éste se añade en la última posición del mismo. Esto conforma el histórico ya que las referencias a los nodos se disponen en el árbol una debajo de otra en el orden en el que fueron visitados. Para acceder a cualquiera de esos nodos, basta con hacer clic sobre su nombre en el árbol. Este árbol puede ser utilizado, desde el punto de vista del lector, como un "mapa" de la zona conocida del hipertexto.

En los comienzos de la etapa de diseño, también se consideró útil la inclusión de marcadores o *bookmarks* que permitieran al usuario la recuperación de un determinado punto de lectura en cualquier momento. De una forma similar actuarían las etiquetas o campos que, mediante su inclusión, facilitarían al lector la recuperación de la información contenida en el hipertexto. Sin embargo, estas dos funcionalidades finalmente no se han implementado debido a su complejidad y a que están fuera del alcance de los objetivos primeros de esta herramienta. Son, por lo tanto, una de las primeras funcionalidades a incluir en futuras etapas de mejora de la aplicación.

4.3.4 La lógica de negocio

Este bloque se compone del conjunto de funcionalidades que se realizan a nivel interno en la aplicación. En concreto, se debían satisfacer los siguientes objetivos:

- Ejecución de las tareas ordenadas por el usuario como respuesta a los eventos generados en la interfaz.

- Tareas auxiliares de la aplicación, entre las que se encuentran principalmente la persistencia de los datos en disco.

Por supuesto, motivado por el hecho de que la exportación de las obras hipertextuales desde HyperAuthor se lleva a cabo basándose en documentos XML, la herramienta de visualización debe ser capaz también de entender este tipo de documentos para poder procesarlos y mostrarlos al usuario. Por lo tanto, una de las tareas principales, de la lógica de negocio de HyperViewer consiste en la traducción de los documentos XML para convertirlos en objetos de información que la interfaz pueda mostrar.

Para este propósito, siguiendo una vez más aquellas directrices adoptadas en el diseño de la herramienta creadora, DOM es la alternativa tecnológica empleada. La recuperación de los archivos almacenados en disco utilizando DOM consiste únicamente en la creación de un árbol, similar al que se crea en HyperAuthor cuando el documento va a ser salvado, que contenga toda la información sobre los nodos contenida en el documento XML. Una vez creado el árbol en memoria, DOM proporciona un conjunto de clases optimizadas que permiten la navegación por él y el acceso a cualquier información que contenga. En resumen, la utilización de DOM en este bloque de la aplicación

simplifica considerablemente la implementación del mismo y además proporciona ventajas en cuanto a posibles compatibilidades y separación del código.

No hay que olvidar, llegados a este punto, que los documentos no son exportados de la herramienta creadora directamente en formato XML. Esto era debido a que además era necesario exportar junto con este documento las posibles imágenes contenidas en los nodos así como la información sobre los estilos aplicados al texto. La solución adoptada en HyperAuthor fue la creación de una carpeta que contuviera toda esta información necesaria para poder recuperar la totalidad de la obra, comprimirla posteriormente utilizando zip y darle la extensión .htxt (ver 4.2.4). Es obvio, por lo tanto, que HyperViewer no sólo debe implementar DOM para poder entender el documento XML, sino que además debe ser capaz de descomprimir la carpeta contenedora de toda la información necesaria para poder reproducir la obra; debe implementar también un descompresor de zip.

Del mismo modo es interesante recordar que la información guardada en el archivo XML sobre el contenido de los nodos consiste en un texto en formato HTML. Para la visualización de este contenido, como se ha comentado en el apartado referido a la interfaz gráfica, se deben borrar primero las marcas que distinguen los enlaces del resto de palabras en el texto para después resaltarlas una vez el usuario haya hecho clic en la zona central del texto.

Esto implica una capacidad de procesamiento por parte de la lógica de negocio que ya no sólo debe ser capaz de entender texto en formato HTML para su presentación en pantalla, sino que además debe ser capaz de escribirlo y modificarlo para poder "marcar y desmarcar" los enlaces a petición del usuario. Por lo tanto este módulo usa, además de DOM, las librerías específicas de Java para el tratamiento y manejo de texto HTML.

Capítulo 5: Implementación

5.1 Introducción

En este capítulo se va a llevar a cabo un análisis de la implementación tanto de HyperAuthor como de HyperViewer. Este análisis se concretará para las dos herramientas de forma separada, y en ambos casos se describirán cada uno de los bloques descritos en el capítulo de diseño.

Se expondrá la estructura de clases generada, señalando las bibliotecas externas empleadas cuando corresponda, así como los puntos más significativos de la dinámica de cada una de las aplicaciones desarrolladas para este proyecto.

El objetivo fundamental de este capítulo consiste en explicar la arquitectura de las aplicaciones realizadas, su estructuración en módulos o paquetes, aquéllas estructuras de datos que sean fundamentales así como los desarrollos algorítmicos no triviales. Con toda esta información se pretende ofrecer al lector un "plano" que le permita obtener una visión general de los programas, así como explicaciones a un nivel más bajo de detalle de aquellos puntos del programa que no resulten obvios.

Se comienza de nuevo por la descripción de HyperAuthor, ya que es la herramienta más compleja de las desarrolladas y en la que recae la mayor parte del peso del proyecto. HyperViewer sigue, en cuanto a arquitectura e implementación software, una estructura similar a la de la herramienta creadora, aunque de una forma más simple; por lo tanto, la descripción de HyperViewer se llevará a cabo en segundo lugar.

5.2 HyperAuthor

Siguiendo el orden establecido en el capítulo de diseño, se comienza por la exposición detallada del proceso de implementación seguido durante la creación de HyperAuthor. De nuevo se utiliza la división en dos grandes bloques (interfaz gráfica y lógica de negocio) para abordar la explicación.

5.2.1 La interfaz gráfica

En el siguiente diagrama UML se muestran las clases más significativas en la estructura software del módulo de interfaz gráfica. Para una referencia completa, el lector puede consultar el Pliego II de este documento: Planos.

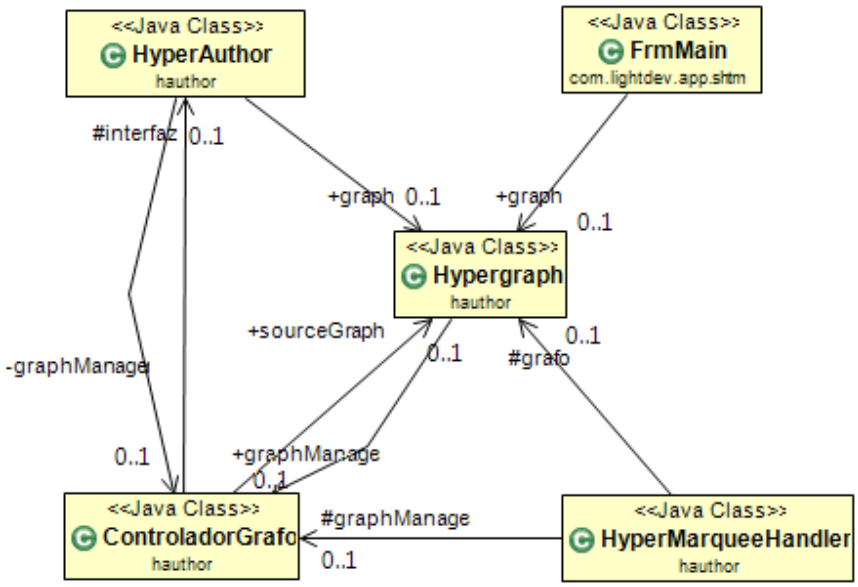


Figura 5.1 Clase principales en la interfaz de usuario de HyperAuthor

Toda la funcionalidad desarrollada para esta herramienta se encuentra dentro del paquete **hauthor**. La clase central del paquete (y de la aplicación completa) es *HyperAuthor*. Esta clase contiene el método *main* y se encarga de presentar la interfaz gráfica en pantalla así como de gestionar los flujos de interacción con el usuario proporcionando:

- Los controles que el usuario puede accionar (botones, menús, teclas de acceso rápido, menús emergentes…).
- El soporte necesario para otros objetos que necesitan residir en un contenedor gráfico.
- Los mecanismos de realimentación hacia el usuario (cuadros de diálogo, mensajes de error, etc).
- Todas las acciones que el usuario es capaz de accionar y sus correspondientes condiciones de habilitación y deshabilitación para poder actualizar su estado de forma dinámica durante el uso de la herramienta.

En resumen, se podría decir que *HyperAuthor* implementa lo que se ha denominado anteriormente "interfaz de usuario" (ver Figura 4.1), que se ajusta a lo descrito en los puntos B y C del apartado 4.2.3. En la siguiente figura se muestra una vista de la interfaz de usuario que proporciona esta clase.

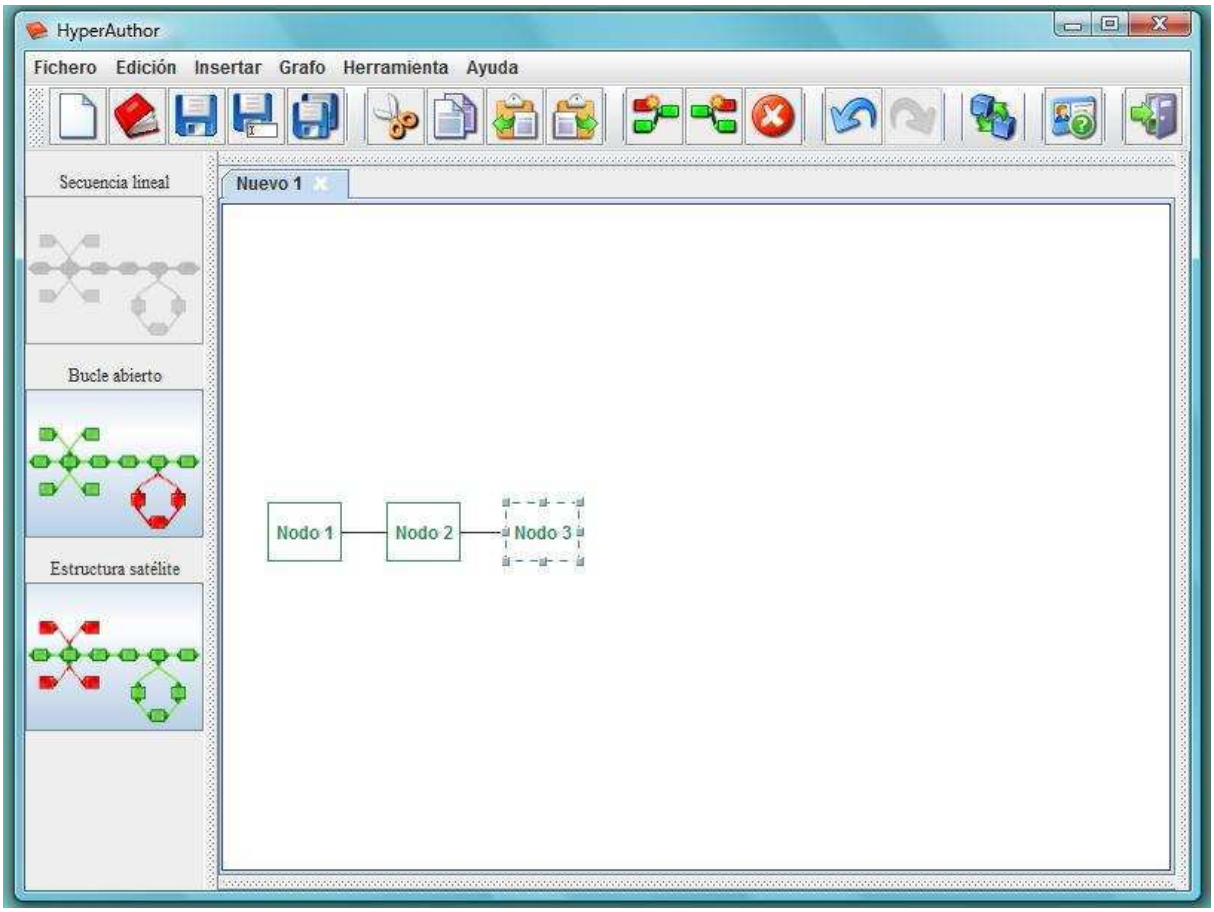


Figura 5.2 Interfaz de usuario de HyperAuthor.

Siguiendo los preceptos establecidos en 4.2.3, HyperAuthor se diseñó a semejanza de un editor de archivos común. Todas las operaciones que el usuario puede realizar en cada momento de la ejecución se encuentran disponibles en la barra de menú. Adicionalmente, y de forma redundante las operaciones que se llevan a cabo de forma más frecuente pueden ser también accedidas desde la barra de herramientas (que puede observarse en la figura), así como a través de un menú emergente activado mediante el botón secundario del ratón y de un conjunto de teclas de acceso rápido predefinidas.

En el panel izquierdo de la aplicación se puede observar aquellas estructuras que conforman lo que se ha estado denominando conjuntos de "estructuras tipo básicas", y que constituyen aquellos tipos de secuencia con las que el usuario cuenta para poder construir la estructura de su hipertexto. Para llevar esta operación a cabo, simplemente se debe hacer clic sobre cualquiera de las estructuras disponibles en un determinado momento e insertar en el cuadro de diálogo que aparecerá en ese momento, el número de nodos del que se quiere que esté formada la nueva secuencia. La zona central es la zona de trabajo, y es empleada para la construcción de la estructura hipertextual.

Cuando se quiere acceder al diseño en el plano de contenido, basta con hacer doble clic sobre cualquiera de los nodos que se hallen en el área de trabajo y se abrirá el editor de nodos, cuya interfaz se muestra en la Figura 5.3 y cuya definición está contenida en la clase *FrmMain*.

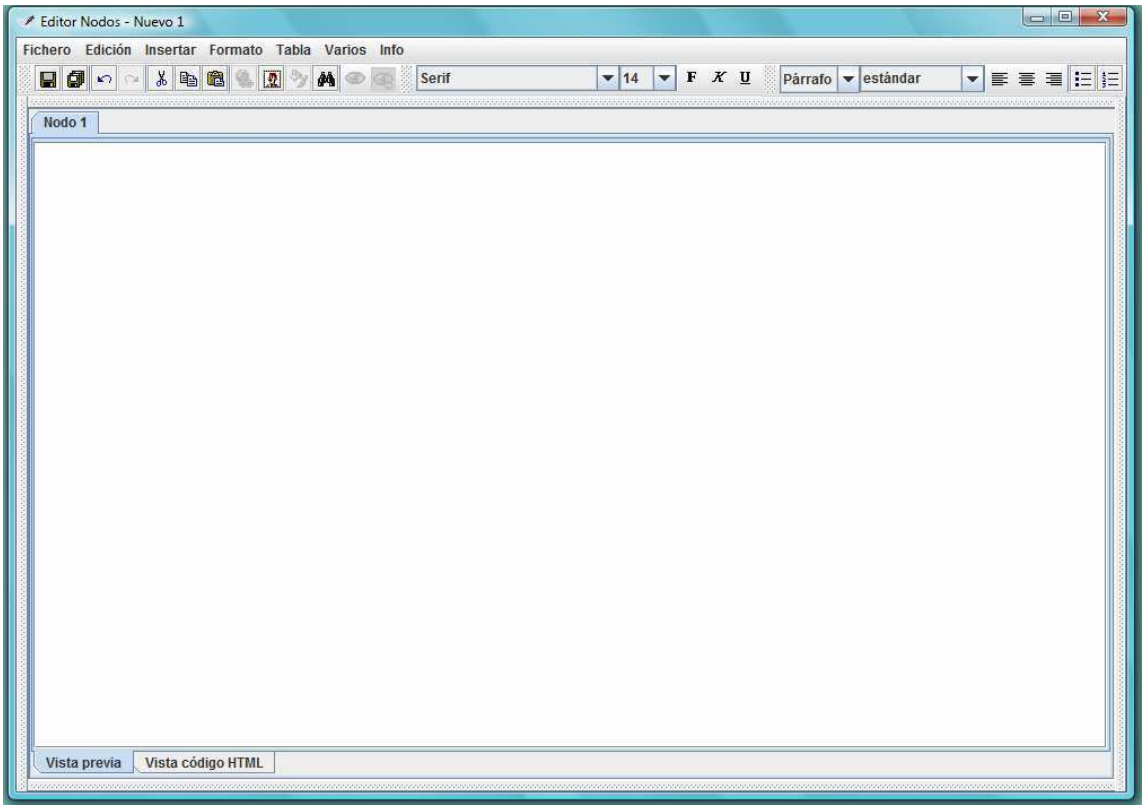


Figura 5.3 Interfaz de usuario del editor de nodos

Ésta es la interfaz del editor de nodos que, como ya se ha comentado, no se ha desarrollado desde el comienzo para los propósitos del proyecto, sino que consiste en la adaptación e integración de un editor de texto ya existente con el código propio de la herramienta. Este editor de texto se denomina SimplyHTML (ver [34]) y está compuesto por varios paquetes diferentes. Estos paquetes son:

- com.lightdev.app.shtm
- com.sun.demo
- de.calcom.cclib.text

La clase *FrmMain* está contenida en el primero de los paquetes, y en cuanto a funcionalidad, se puede decir que es la clase análoga a HyperAuthor dentro del editor. Esta clase es la encargada de construir la interfaz gráfica así como de recibir y gestionar los eventos generados por el usuario en su interacción con la misma. Para conocer más detalles sobre la aplicación en concreto, el lector puede dirigirse a la documentación oficial en [34]. En lo relativo a las adaptaciones necesarias llevadas a cabo en la aplicación, éstas

han sido ya descritas en el capítulo 4 y en ningún caso han implicado una modificación en la estructura de clases del editor original.

A partir de este momento del documento, el editor de nodos será tratado como un todo, motivo por el cual no se va a hacer una descripción de su estructura interna. Sin embargo, sí se expondrán, si así se estima oportuno, aquellos aspectos relativos a la implementación del código de las adaptaciones que puedan resultar interesantes y aportar riqueza a este documento.

Como se puede apreciar en la figura 5.3 y al igual que ocurre con la interfaz de HyperAuthor, todas las operaciones que puede realizar el usuario se encuentran disponibles en la barra de menús, y además algunas de ellas también desde la barra de herramientas y desde teclas de acceso rápido predefinidas. Las funcionalidades específicas añadidas a esta interfaz son aquellas relacionadas con los enlaces: crear/editar enlaces y activación /desactivación de los mismos. De nuevo el área de trabajo ocupa el área central de la ventana. En ella se presentan al usuario los documentos con los que se está trabajando y se permite su edición.

En este punto no merece la pena llevar a cabo una exposición más detallada del proceso de implementación ya que, al margen de lo comentado sobre el uso del editor de texto, se trata de una tediosa sucesión de declaraciones de paneles, menús, botones, escuchadores y códigos de acción, etc.

Por otro lado, el lector interesado en una descripción más exhaustiva de la funcionalidad de la interfaz de usuario, a más alto nivel, puede consultar el "Manual de Usuario" que se encuentra en el pliego III de este proyecto.

Sin embargo, como consecuencia de ejercer como clase principal de la aplicación, *HyperAuthor* se responsabiliza, como respuesta ante la gestión de los eventos generados por las acciones del usuario, del control del flujo sobre el resto de clases. Este punto resulta de especial relevancia para la comprensión del conjunto. Cabe resaltar también que, cuando el editor de nodos se encuentra en uso, la clase *FrmMain* toma el relevo y es ella quien gestiona los eventos generados por su propia interfaz hasta que el control vuelve a *HyperAuthor*.

HyperAuthor permite manejar varios diagramas en el área central de trabajo de forma simultánea. Esto se lleva a cabo mediante el uso de un clasificador del tipo *JTabbedPane* [31]. En cada instante, esta clase se encarga de mantener el diagrama que aparece en la pestaña seleccionada como "activo", lo que quiere decir que será el diagrama que reciba todas las acciones realizadas por el usuario. Cada diagrama es un objeto del tipo *Hypergraph*. *Hypergraph* es sencillamente una clase que implementa un grafo completamente editable, con las limitaciones de formato que exige la herramienta, descritas en el apartado 4.2.3 D. Contiene también la codificación de algunos métodos útiles y necesarios para moverse por la estructura del grafo o para acceder a algunas de sus propiedades. Como se observa en el diagrama de clases, *Hypergraph* hereda de la clase *JGraph*, que pertenece a una librería que ya se ha nombrado en este documento: la librería Jgraph [33].

Jgraph es una librería externa de código abierto y muy versátil que se adecua muy bien a las necesidades de la herramienta, razón por la que se decidió emplearla. De un modo resumido, y en palabras similares a la descripción que proporciona la documentación de sus clases, se puede decir que Jgraph proporciona un objeto gráfico complejo que presenta información siguiendo el paradigma de un grafo. Proporciona también un amplio conjunto de clases auxiliares necesarias para implementar ese paradigma dentro del entorno gráfico facilitado por Swing [31].

La gran ventaja que suponía para el desarrollo de este proyecto la utilización de Jgraph frente a la implementación de objetos gráficos personalizados quedó patente en el momento de estudiar la documentación de la librería [33]: la clase *JGraph* encapsula toda la complejidad de visualización del grafo, convirtiéndolo en un componente Swing más, que se puede situar en cualquier contenedor como si de un simple botón o etiqueta se tratase. Por ejemplo:

- Gestiona el dibujo y refresco de sus nodos y ejes.

- Permita el tratamiento del dinamismo en la posición y tamaño de los nodos, que se puede redimensionar y mover al gusto sobre la superficie disponible manteniendo siempre las conexiones existentes (la librería se encarga del repintado y refresco en pantalla de nodos y ejes de forma automática).

- Gestiona automáticamente su propio tamaño al ir añadiendo nodos nuevos. Aunque esta funcionalidad dejó de ser necesaria cuando, en la fase de diseño, se decidió incorporar un scroll al área de trabajo, se contempló como una ventaja interesante en los inicios de la herramienta.

- Permite la selección de celdas y la transferencia de objetos gráficos como si de cualquier componente se tratara, tanto dentro de un mismo grafo como entre dos grafos diferentes.

De modo resumido se puede decir que Jgraph es capaz de proporcionar, de una forma simple, la componente de interacción requerida durante el proceso de construcción de la estructura hipertextual. Por este motivo, resultaba el soporte ideal para el grafo con las características planteadas en el apartado 4.2.3 A.

Sin embargo, aunque la funcionalidad ofrecida por *JGraph* en lo relativo al aspecto visual cubría con creces las necesidades que HyperAuthor planteaba, no ocurría lo mismo en lo concerniente a los datos que el diagrama debía ser capaz de contener. En este sentido se identificaron algunas carencias: los datos pertenecientes a la dimensión de contenido deben poder estar incluidos en los objetos gráficos, de modo que la interfaz gráfica sea capaz de llevar a cabo todas las tareas que se le asignaron en la fase de diseño. Por este motivo, se planteó en este punto la necesidad de implementar unas funcionalidades más específicas de las que se podían obtener con el uso de de las clases de la librería.

Como todo componente Swing, *JGraph* está diseñado según la arquitectura *"separable model architecture"* (ver [35]) basada en el paradigma MVC, el mismo usado tanto para la implementación de HyperAuthor como de HyperViewer, y cuya implicación fundamental es la separación entre los datos que el control necesita para funcionar como tal (modelo), la lógica de pintado del componente (vista) y la lógica de control entendida como el comportamiento del componente ante las acciones del usuario sobre él (controlador). En la arquitectura desarrollada por Swing, la vista y el controlador se fusionan y la división queda reducida a modelo y UI (objeto que encapsula los dos últimos).

En el caso de HyperAuthor, el objeto UI de *JGraph* podía ser usado de forma inalterada, pero no así el modelo, ya que éste determinaba los datos que el grafo debía contener. Por esta razón se decidió crear la clase *HyperGraph*, heredando de su clase correspondiente en la librería. En los siguientes párrafos se encuentran los detalles al respecto.

La creación de la clase *HyperGraph* obedece a los siguientes motivos:

- Añadir ciertas propiedades de instancia que un grafo creado para funcionar como estructura hipertextual debe contener para su operación dentro de la aplicación: un nombre, una referencia al objeto *ControladorGrafo* de la aplicación, etc.

- Añadir métodos que envolviesen el manejo del modelo del grafo facilitando ciertas operaciones comunes sobre los grafos dentro de HyperAuthor. Por ejemplo, un método para proporcionar el nodo siguiente a un nodo determinado dentro del grafo, o un método para el primer nodo de todos y cada uno de los bucles que tienen como origen un determinado nodo. El primero de los métodos, junto con su gemelo que devuelve el nodo anterior a uno dado asocian de alguna manera al grafo un orden de recorrido que no tiene de una manera inherente. El segundo método se proporciona para facilitar la serialización del objeto HyperGraph así como para ayudar al repintado y validación del mismo durante el uso de la herramienta.

- Añadir al grafo la capacidad de tener una historia de eventos de deshacer y rehacer propia. De esta manera no existe un historial de eventos común en la herramienta para todos los grafos abiertos, sino muchas pilas de eventos diferentes, una para cada uno de los grafos.

Un punto importante a destacar en este momento es que *JGraph* establece que la información contenida en cada nodo y eje del grafo es almacenada por el modelo en una propiedad denominada *userObject*. *DefaultGraphModel*, que es el modelo que implementa Jgraph por defecto y que se ha usado como tal en la herramienta, no proporciona funcionalidad adicional en cuanto al tratamiento de ésta información, sino que almacena en las celdas (ejes o nodos) objetos genéricos. Sin embargo, en el caso de HyperAuthor los nodos deben actuar como contenedores de toda aquella información relativa a la dimensión de contenido y de la cual es necesario disponer, tanto a nivel de usuario como a

nivel de grafo. De esta información dependen aspectos como, por ejemplo, el texto que se debe mostrar en las etiquetas tanto de los nodos como de los ejes. Precisamente para suplir esta necesidad se creó la clase *Nodo*, cuya descripción se encuentra en la siguiente figura.

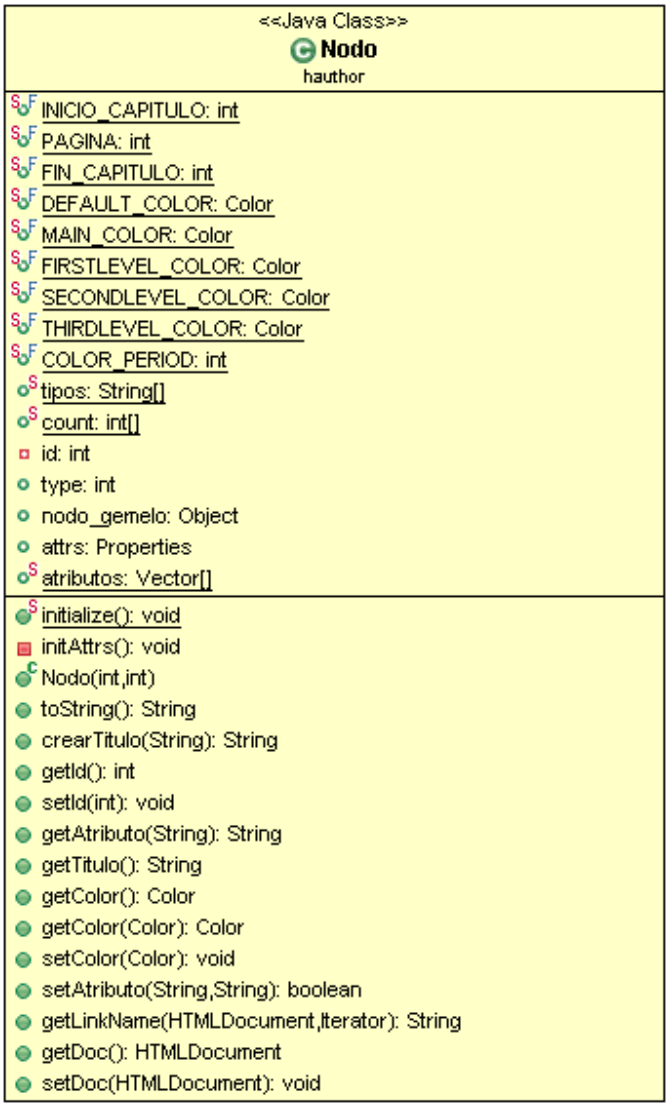


Figura 5.4 Vista UML de la clase *Nodo* empleada en HyperAuthor.

Este objeto *Nodo* se emplea como soporte para almacenar toda aquella información necesaria del mismo perteneciente a la dimensión de contenido, así como alguna información necesaria para su correspondiente representación en pantalla (por ejemplo, el color) e incluso un soporte para la implementación de diferentes tipos de nodo, pensada para crear nodos de inicio de capítulo, de final de capítulo, o nodos página (aquellos que no pertenecen a ninguno de los dos tipos anteriores) que finalmente no se utiliza en la versión actual de la herramienta. Estos objetos *Nodo* son los que se almacenarán en los nodos que forman el grafo sin que el modelo conozca de su existencia.

Es también importante mencionar que los objetos *Nodo* se emplean como objetos de intercambio con las clases pertenecientes a la lógica de negocio. Esta relación se verá más adelante en el punto 5.2.2.

Como se puede intuir a partir del diagrama de la figura 5.4, la clase *Nodo* lleva gran parte del peso de gestión de los nodos. En concreto:

- Mantiene los distintos tipos de colores disponibles en la aplicación en los que puede ser representado un nodo atendiendo a su profundidad en el grafo. Con este fin se usan constantes de clase públicas.

- A pesar de que este mecanismo no está en uso por el momento, mantiene también los distintos tipos de nodos disponibles mediante la utilización, de nuevo, de constantes de clase públicas.

- Cada instancia de clase contiene atributos como el identificador, el título o el texto que permite distinguir unos nodos de otros. Estos atributos son generados automáticamente por el constructor de la clase, que siempre recibirá como parámetros el identificador de nodo (que debe ser único dentro de cada grafo) y el tipo que debe tener la instancia. Se iniciarán con sus valores por defecto, que deberán ser modificados por el usuario durante el momento de construcción de la obra.

De un modo más intuitivo, se puede decir que un objeto *Nodo* es la "esencia" de un nodo perteneciente al diagrama, ya que contiene todos aquellos datos que la particularizan, distinguiéndola de cualquier otra instancia; y lo es tanto en el ámbito de la interfaz gráfica como en el de la lógica de negocio, que se estudiará en el siguiente apartado.

La clase dentro de la librería Jgraph que representa un nodo a nivel de grafo es *DefaultGraphCell* [33]. Las instancias de esta clase son objetos genéricos con propiedades genéricas, que contienen una propiedad *userObject* del tipo *Object* que les permite almacenar información. Tal y como se ha comentado con anterioridad, en HyperAuthor el modelo del grafo almacenará una instancia de la clase *Nodo* dentro de cada *DefaultGraphCell*, rellenando de esta manera el atributo *userObject.* Una vez almacenado el nodo correspondiente, éste convierte al *DefaultGraphCell* en un nodo con propiedades concretas tanto en la dimensión estructural como en la dimensión de contenido. El proceso es como verter una gota de tinte en un algodón (de repente todo el algodón se pone del color del tinte) y se hace a dos niveles:

- Queda almacenada en el nodo la información relativa a la dimensión de contenido que el usuario debe tener accesible cuando así lo requiera y que posteriormente se pasará a la lógica de negocio para el mantenimiento de la misma.

- El aspecto visual del nodo, en términos de color, queda determinado. Cada nodo tendrá un color distinto dependiendo de la profundidad a la que se

encuentre, y este color será modificado si la posición relativa del nodo dentro del grafo también lo es.

En principio, a la vista de todo lo comentado hasta el momento, *DefaultGraphCell* es capaz de aportar toda la funcionalidad necesaria para la funcionalidad de la herramienta. Sin embargo, en una última fase de depuración de la herramienta se decidió crear la clase *HyperGraphCell*, que hereda de *DefaultGraphCell* y que aporta una funcionalidad extra que se creyó interesante incluir.

Ya es conocido que los nodos existentes dentro del grafo muestran, mediante una etiqueta en su interior, el título del nodo que contienen. De esta forma, el autor puede distinguir unos de otros incluso trabajando en la dimensión estructural. Cuando un grafo alcanza una complejidad, y por lo tanto también un tamaño, dignos de mención, es posible que el título asignado a un nodo no pueda ser mostrado por completo en su interior porque no exista espacio disponible. En ese momento se pensó que sería una buena idea que, al colocar el ratón sobre un nodo, su título apareciese completo en la típica etiqueta amarilla que utilizan los tooltip de los menús o algo similar.

La librería Jgraph contemplaba el uso de esta funcionalidad, y proporcionaba mecanismos para ello; por este motivo la funcionalidad quedó implementada en la versión definitiva de la herramienta. Sin embargo, el método de implementación que aparecía en la documentación (ver [36]) exigía de varias acciones diferentes:

- Registrar el objeto grafo con el *ToolTipManager.*

- Implementar una clase que heredara de *DefaultGraphCell* y crear en ella el método *getToolTipString()* que devolviera el texto a mostrar en la etiqueta cuando el ratón estuviera sobre el nodo.

- Implementar una clase que heredara de *JGraph* y crear en ella el método *getToolTipText()* que recibe un evento de ratón, accede a la celda en la posición determinada por el evento y devuelve el string que la celda devuelve a modo de tooltip.

En este momento de la implementación ya existía la clase *HyperGraph,* que hereda de *JGraph* y en la cual se incluyó el método mencionado. Sin embargo, no había sido necesario hasta el momento el uso de una clase que heredara de *DefaultGraphCell*. Es en este instante cuando se crea *HyperGraphCell*, que ya se ha comentado que hereda de *DefaultGraphCell* y cuya misión principal consiste en alojar el método mencionado en las especificaciones anteriores para poder implementar la funcionalidad de "tooltip" en el grafo. El resultado puede apreciarse en la siguiente figura.

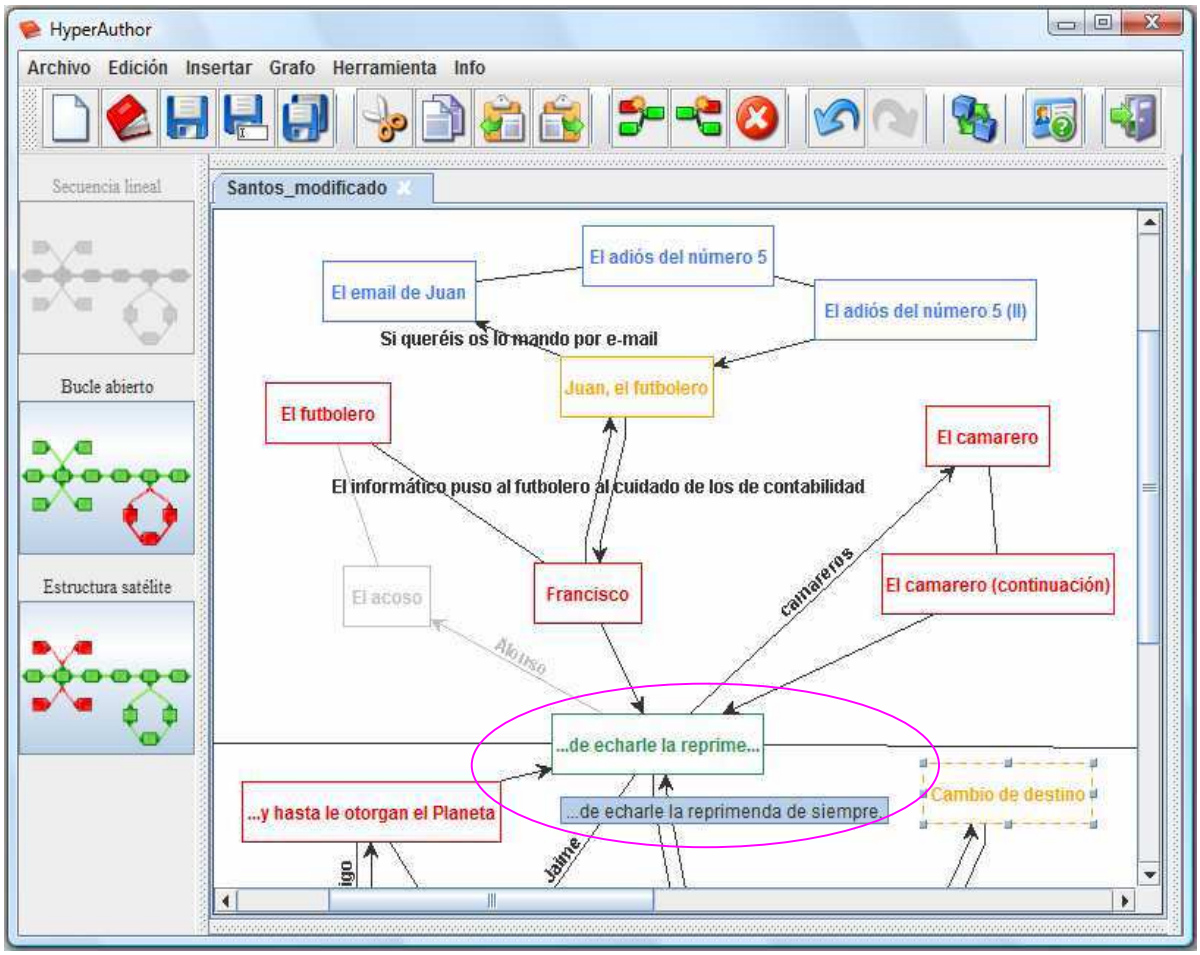


Figura 5.5 Tooltip sobre un nodo en HyperAuthor.

Una vez creada, la clase *HyperGraphCell* se mostró útil también para implementar el método *getNodo()*, un método similar al método *getUserObject()* que ya implementaba la clase padre, pero más específico. Este nuevo método ya no devuelve un *Object* sino directamente un objeto *Nodo*, evitando, de esta forma, la utilización de conversiones explícitas a lo largo de todo el código cuando se quiere acceder al contenido específico de un nodo.

En cuanto a la gestión de la edición de nodos, la herramienta requería proporcionar acceso desde la interfaz de usuario a los atributos de los nodos concretos almacenados en cada celda, es decir, a las propiedades del objeto *Nodo* almacenado en el objeto gráfico que lo representa. Tal y como se ha comentado, la forma más intuitiva desde el punto de vista del usuario consistía en hacer doble clic en el nodo que se quería editar para que se abriera el editor de nodos en el que se pudieran llevar a cabo las operaciones de edición necesarias.

Sin embargo, los grafos ya tenían asociada una operación por defecto como respuesta a un doble clic sobre uno de sus nodos. Esta respuesta consistía en la aparición sobre el nodo de un pequeño cuadro de texto que permite la edición de la etiqueta asociada a él. Para poder modificar esta acción por defecto se creó la clase *HyperMarqueHandler.* Esta clase hereda de *BasicMarqueeHandler* [33], una clase proporcionada por la librería Jgraph que implementa un gestor de eventos de ratón específico para los grafos.

Al heredar de esta clase, *HyperMarqueeHandler* implementa el método *isForceMarquee*(), que permite indicar qué eventos de ratón van a ser manejados directamente por esta clase y cuáles van a ser redirigidos al gestor por defecto. De esta forma, cuando se produce un evento de doble clic, la clase comprueba si se ha llevado a cabo sobre alguno de los nodos contenidos en el grafo, en cuyo caso abrirá el editor de nodos con los atributos actuales del nodo correspondiente.

Si por el contrario se ha hecho doble clic sobre un eje, se lleva a cabo una comprobación sobre el modo de visualización de etiquetas activo en la herramienta en ese momento concreto (ver 4.2.3 G). En el caso de que el modo activo sea el modo semi-visible, el resultado de la acción de doble clic sobre el eje consistirá en que su correspondiente etiqueta será mostrada u ocultada de forma pertinente. En cualquier otro modo de visualización, un doble clic por parte del usuario no obtendrá ninguna respuesta de la herramienta.

De forma adicional esta clase facilita una segunda misión importante: permitir el uso y gestión de los menús emergentes. *HyperMarqueeHandler* es capaz de distinguir cuando se ha pulsado el botón secundario del ratón, y cuando detecta un evento de este tipo y después de discernir si el clic se ha llevado a cabo sobre el grafo o sobre una zona vacía del área de trabajo, invoca a los métodos adecuados de la interfaz de usuario que hacen que se muestren los menús emergentes correspondientes. Este comportamiento puede apreciarse en la siguiente figura.

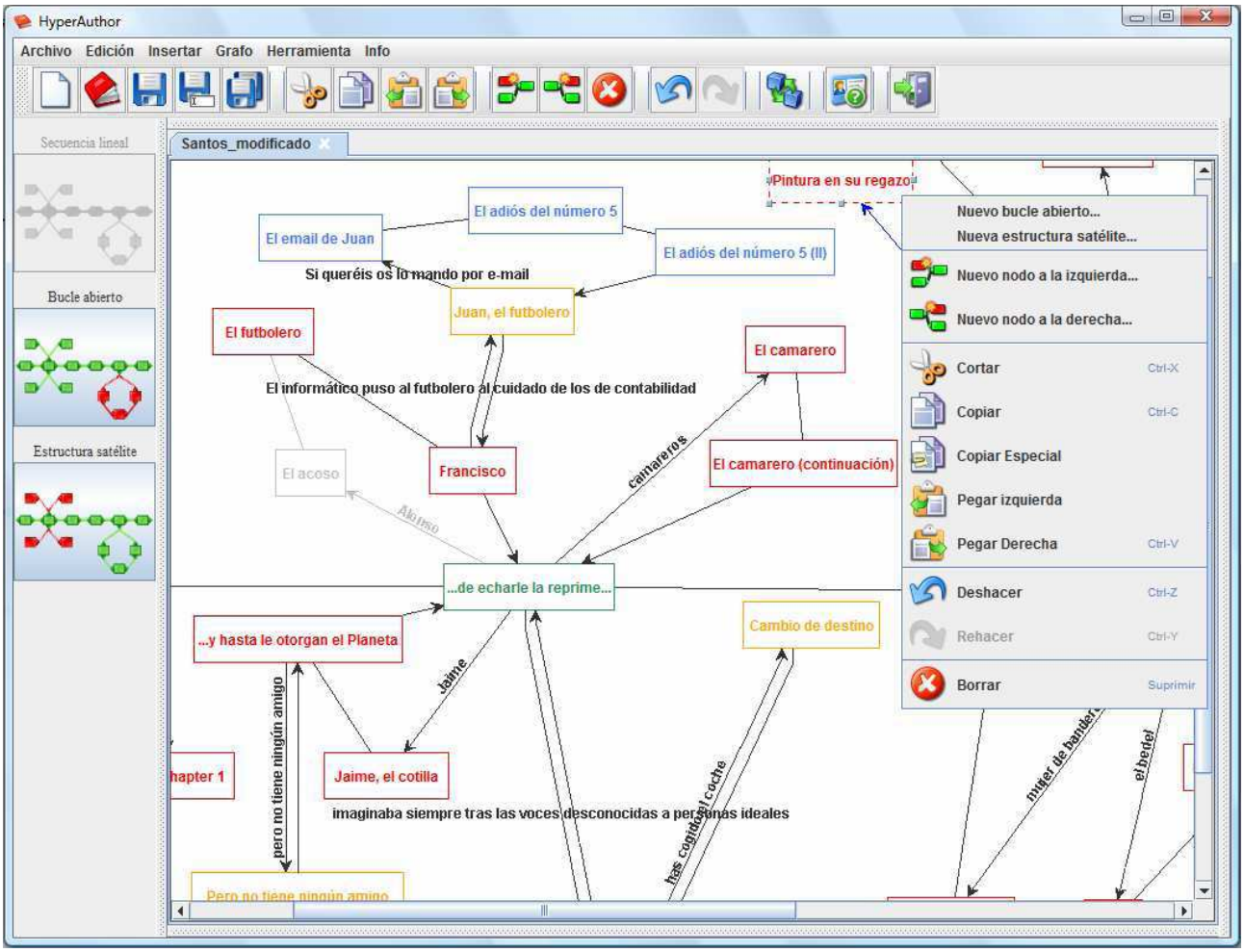


Figura 5.6 Menú emergente de HyperAuthor

Volviendo a la clase *Nodo,* es importante explicar lo que antes se ha mencionado de un modo somero: además de como contenedor de información, esta clase actúa también como objeto de intercambio. Debido a este segundo papel que desempeña, *Nodo* dispone de atributos y métodos relacionados con cada uno de los dos grandes bloques de los que está compuesta la aplicación y en los que se intercambia:

- Para el bloque de la lógica de negocio, *Nodo* dispone del objeto *nodo_gemelo.* En el apartado 5.2.3 se explicará en qué se emplea.

- Para el bloque que implementa la interfaz gráfica, dispone del atributo *color*, que representa el color con el que el nodo será dibujado en pantalla. *Nodo* almacena a nivel de clase el color asociado con cada profundidad que puede tener el nodo en el grafo.

Ambas propiedades de instancia, aunque pertenecientes a mundos distintos, son, sin embargo, propiedades intrínsecas al grafo que maneja la aplicación y que adquieren su sentido cada una en su mundo correspondiente. De este modo, *Nodo* actúa como enlace

entre la interfaz gráfica y la lógica de negocio. Esto implica que cualquiera de los dos bloques puede construir un *Nodo* (por ejemplo, la interfaz al crearlo el usuario desde la herramienta y la lógica de negocio al abrir desde fichero un grafo guardado en disco) y pasarlo al otro. Los dos bloques estarán, por lo tanto, operando con la misma estructura hipertextual concreta (incluyendo su contenido).

Por otro lado, dado que un *nodo_gemelo* es un objeto genérico de tipo *Object* y *color* es simplemente un *String* que representa el identificador del color del nodo, cada uno de los dos bloques dispone de total libertad para implementar el nodo del modo más conveniente para ellos, con independencia del otro bloque. De este modo se refuerza la separación entre los dos bloques de la aplicación que se defendía durante el capítulo dedicado a la fase diseño.

Pero volvamos a lo que nos ocupa en este apartado, la implementación de la interfaz gráfica. El hecho de que el *Nodo* almacene el color con el cual se va a dibujar el nodo no implica que haga nada para pintarla efectivamente del color indicado, se trata simplemente de un objeto contenedor. Aquí es donde entra en juego la clase *GraphConstants*, ofrecida por la librería y que permite acceder a los atributos de los nodos y ejes representados en el grafo mediante lo que en la librería se denomina *AttributeMap,* que no es más que una lista de pares atributo-valor que representan todas las propiedades editables de un nodo o eje.

La clase *GraphConstants* permite modificar mediante los métodos *setBorderColor* y *setForeground* el color con el que se dibujará un determinado nodo en pantalla. Pero es más, debido a que el color de un nodo puede variar durante el ciclo de vida de la aplicación (provocado por un cambio de posición del mismo llevado a cabo, por ejemplo, mediante una acción de cortado y pegado y que implique un cambio también en la profundidad), la información al respecto se almacena en esta clase y no en el objeto *Nodo*.

Esto es debido a los mecanismos de deshacer y rehacer ofrecidos por la herramienta y que afectan de forma exclusiva a aquella información contenida de una forma u otra en el objeto grafo. Esto quiere decir que todas aquellas variables almacenadas en memoria como atributos de clase no se ven afectadas por las acciones de deshacer o rehacer. Por lo tanto, si una acción de cortado implicara un cambio en el color de los nodos cortados y esta información de color se recuperara del atributo color de la clase *Nodo* mencionado anteriormente, una posterior acción de deshacer que pudiera invocar el usuario llevaría a cabo un movimiento de la posición de los nodos afectados pero no un cambio en su color, de forma que el color original se perdería.

La clase *GraphConstants* almacena y actualiza este tipo de información con las acciones deshacer y rehacer, de modo que durante el tiempo que el grafo esté abierto y en uso, será esta clase quien determine las propiedades de los nodos. Una vez el grafo vaya a ser guardado, esta información que mantiene la interfaz gráfica se actualiza en las variables de clase correspondientes para generar objetos *Nodo* que contengan toda la información necesaria para ser exportados a fichero.

Pues bien, después de esta aclaración sobre la vista de los nodos ya sólo resta por exponer lo referente a la última de las clases que aparecen en el diagrama de la figura 5.1: la clase *ControladorGrafo.* Esta clase tiene una triple función:

- Implementa para HyperAuthor el engorroso tratamiento de los objetos *HyperGraph,* escondiendo su complejidad.

- Mantiene información de control sobre los distintos grafos abiertos en la aplicación, como puede ser si existen cambios sin guardar en un determinado grafo, el nombre del fichero donde se guarda, del que ha sido abierto… Sin embargo, probablemente su función más importante sea la de gestionar las diferentes pestañas; ser conocedor de cuál es el grafo activo en cada momento.

- Centraliza las operaciones de intercambio de información con la lógica de negocio cuando este intercambio es necesario para dar respuestas a las acciones ordenadas por el usuario en *HyperAuthor.*

En cuanto al primer punto, *ControladorGrafo* proporciona métodos para añadir o eliminar secuencias a un grafo, así como para copiar, cortar y pegar elementos, ya sea dentro de un grafo o entre un grafo y otro. Todas las acciones ser realizan, por supuesto, sobre el grafo activo, que *ControladorGrafo* mantiene referenciado en su variable *grafo*. La clase *HyperAuthor* será responsable en cada momento de notificar al *ControladorGrafo* el cambio de grafo activo para que éste pueda operar de forma correcta.

Para añadir una secuencia nueva, *HyperAuthor* invoca el método correspondiente de *ControladorGrafo* pasándole como parámetros el número de nodos que formarán la nueva secuencia y, en las secuencias de bifurcación, también el nodo origen y destino de la misma, que será el nodo que se encuentre seleccionado en ese momento. *ControladorGrafo* creará un objeto *Nodo* para añadírselo a cada objeto *HyperGraphCell* que deba integrar en el grafo, pasándole como parámetros al constructor un identificador único y el tipo (ya se ha comentado que de momento la funcionalidad asociada al tipo de nodo no está en uso).

La secuencia se añadirá al grafo, creándose también las conexiones necesarias con el nodo origen y destino de la secuencia en el caso de que se trate de un bucle abierto o una estructura tipo satélite. En cuanto a la eliminación, también debe llevarse a cabo de una forma que no comprometa el cumplimiento por parte del grafo de todas las reglas expuestas en 4.2.3.A. Por este motivo, antes de llevar a cabo la supresión definitiva de los nodos o cualquier acción de copiado o cortado, *ControladorGrafo* comprueba:

- Si los nodos seleccionados son consecutivos. Esta operación se lleva a cabo mediante el método *areConsecutive().*

- En caso de que los nodos pertenezcan a secuencias no lineales, mediante el método *isLoopComplete()* comprueba si todos los nodos pertenecientes a esa estructura están también seleccionados.

- Por último, comprueba que junto con el nodo origen, se copien todas las secuencias que lo tienen como origen mediante el uso del método *isStructureComplete().*

Para las operaciones de copiar y pegar, *ControladorGrafo* emplea un objeto *GraphTransferHandler* [33], al que copia los elementos seleccionados en el grafo activo. Una vez se realiza la acción de pegado, en la cual se importan los datos del mismo objeto *GraphTransferHandler*, esta clase se encarga de posicionar los nuevos nodos de una manera adecuada con respecto al nodo que se tomó como referencia y de reajustar los enlaces contenidos en el grafo para que éste sigue siendo válido.

Entre las responsabilidades de este objeto se encuentran también, además de las mencionadas, las de implementar las acciones de deshacer y rehacer sobre los grafos así como las de gestionar la edición de nodos, de construir el grafo cuando se abre una obra desde fichero o el de guardarlo cuando la obra vaya a ser salvada.

Las acciones de deshacer y rehacer se implementan con ayuda de la clase *GraphUndoManager* de Jgraph [33]. Esta clase incluida en la librería proporciona la funcionalidad de deshacer o rehacer los cambios realizados sobre el objeto *JGraph.* Es importante hacer notar que esta capacidad de deshacer y rehacer afecta de forma única y exclusiva a aquellas acciones realizadas sobre el grafo, y que, por lo tanto, no afectan a cualquier otro tipo de objetos. Por este motivo, la información almacenada, por ejemplo, en el objeto *Nodo* alojado dentro de una determinada celda no se ve afectada por estas acciones.

Este hecho implica que, si se quiere que la aplicación ofrezca la posibilidad de deshacer y rehacer al usuario, la interfaz gráfica, y más concretamente el objeto *HyperGraph* y los objetos *HyperGraphCell* que lo componen, deben ser capaces de mantener y gestionar toda la información relativa a la dimensión estructural de la obra sin ayuda de variables almacenadas en memoria. De otra forma, las acciones rehacer y deshacer no tienen ningún efecto y el resultado de las mismas podría resultar en un grafo no válido.

El *GraphUndoManager* almacena los objetos *UndoableEdit* que contienen los eventos *UndoableEditEvent* producidos por los grafos en su calidad de componentes Swing y gestiona la pila proporcionando métodos que permiten aplicarlos o retrocederlos. Esta clase, como puede comprobarse en la documentación [33], implementa la interfaz Swing *UndoableEditListener*, de modo que en un principio, permitiría ser registrada directamente en los grafos para encargarse automáticamente, y sin necesidad de crear código nuevo, de estos eventos.

Durante la implementación de la herramienta se encontró una pequeña carencia a este respecto. Tal y como se ha comentado, uno de los requisitos de la herramienta consiste en ofrecer al usuario una herramienta atrayente, simple y moderna; parte de este requisito reside en el hecho de que los menús y barras de herramientas se habiliten y deshabiliten de manera dinámica dependiendo de si el usuario puedo o no ejecutar esa acción en un momento determinado. Las acciones deshacer y rehacer, de manera similar al resto de las acciones implementadas en HyperAuthor, son ejecutadas como respuesta a un evento

generado por el usuario en su interrelación con la interfaz. Por lo tanto, los menús, botones y demás elementos asociados a las acciones de deshacer y rehacer deben ser capaces de deshabilitarse cuando la pila de *UndoableEdit´s* no permita su ejecución.

Sin embargo, dejando la gestión de este tipo de eventos directamente en manos de *GraphUndoManager* no se podía obtener ese comportamiento, ya que estos eventos eran manejados de forma transparente. Para solucionarlo, por un lado se utilizaron las clases *RedoAction* y *UndoAction* contenidas en *HyperAuthor* y, por otro, se definió la clase *UndoHandler* como clase interna dentro de la clase *ControladorGrafo*.

Las dos primeras heredan de *AbstractAction* y, principalmente, se ocupan de actualizar su estado de habilitación cada vez que se produce un evento. Si tras un evento detectan que no existen *UndoEdit´s* adecuados para ellas en la pila, lo que averiguan a través de la comunicación con el objeto *GraphUndoManager,* se deshabilitarán.

Por su lado, la clase *UndoHandler* se ocupa de interceptar los eventos correspondientes y redirigirlos al *GraphUndoManager* del grafo adecuado. Por este motivo la clase *ControladorGrafo* dispone en cada momento de una única y exclusiva instancia de *UndoHandler* mientras que, al mismo tiempo, cada *Hypergraph* contiene una instancia de su propio *GraphUndoManager*. Esto permite que cada grafo creado en la herramienta pueda tener una pila de eventos propia.

Una vez implementada las funciones de deshacer y rehacer del modo descrito anteriormente, se detectó que había que dar un paso más en la definición de estas acciones. Hasta ahora sólo se había pensado en *UndoEdit* como unidad mínima de edición, sin embargo en la herramienta se implementan acciones que generan no sólo uno, sino varios de estos eventos de undo. Por ejemplo, una acción de pegado tras una acción de cortado lleva implícitas las siguientes acciones: una acción de eliminación de los nodos que aparecen marcados como cortados en el área de trabajo así como una acción de inserción de las nuevos nodos y una última acción de reubicación de los mismo. Cada una de estas acciones genera su propio *UndoEdit.* De forma obvia, cuando el usuario pulsa deshacer después de haber llevado a cabo esta acción de cortado, lo que espera es que se deshagan cada uno de los *UndoEdit* mencionados, y no sólo uno de ellos, lo que provocaría que el diagrama quedara en un estado inconsistente.

Por lo tanto, la implementación de las acciones de deshacer y rehacer no podía llevarse a cabo en el modo que se ha mencionado con anterioridad. Para solucionar esta situación, se definió dentro de la clase *UndoHandler* una variable de tipo *CompoundEdit* [31]. Ésta es una clase Swing que hereda de *AbstractUndoableEdit* y cuya finalidad consiste en unir pequeños *UndoableEdit´s* en un único evento mayor de forma que todos esos pequeños eventos agrupados puedan ser usados de forma atómica por la herramienta.

Con este mismo propósito se definieron, dentro de *ControladorGrafo,* la variable de clase *compoundInProgress*, y los métodos *startCompoundAction()* y *endCompoundAction().* La primera consiste en un booleano utilizado para poder diferenciar internamente si la acción que se está llevando en la herramienta es o no una acción compuesta desde el punto de vista de eventos de deshacer. Lo que es lo mismo, esta variable permite saber si una acción

en concreto va a generar o no más de un *UndoEdit*. Atendiendo al valor de esta variable, el objeto *UndoHandler* añade el evento generado directamente al *GraphUndoManager* del grafo correspondiente o por el contrario, lo añade al *CompoundEdit* que tiene como variable de clase.

En cuanto a los métodos *startCompoundAction()* y *endCompoundAction()* se invocan por las acciones compuestas antes y después de ejecutar las tareas necesarias para llevar a cabo su función. El primero de los métodos se encarga simplemente de poner a "true" el valor de *compoundInProgress*, lo que provoca que los eventos generados a partir de ese momento se agrupen en un único evento compuesto. El segundo de los métodos se encarga de cerrar este evento compuesto (que ya contiene todos los eventos simples generados por la acción), y de añadirlo a la pila del *GraphUndoManager* adecuado. De esta manera queda implementada definitivamente la funcionalidad de deshacer y rehacer.

ControladorGrafo implementa también la interfaz *GraphSelectionListener* [33]. Un objeto que implementa esta interfaz recibe los cambios producidos en el conjunto de elementos seleccionados en los grafos a los que escucha. El objetivo de que *ControladorGrafo* implemente esta interfaz reside básicamente en poder actualizar la barra de herramientas y los menús de HyperAuthor cada vez que la selección en el grafo en uso varíe. De forma adicional, también controla que los ejes no puedan ser seleccionados de forma aislada, ya que no se puede hacer nada con ellos. Si un eje es añadido a la selección y ni su origen ni su destino se encuentran también seleccionados, *ControladorGrafo* lo elimina de la selección para que el usuario no tenga la falsa percepción de que puede trabajar con él.

La única excepción a esta regla se hace cuando la herramienta tiene configurado un modo para las etiquetas de los ejes distinto al modo invisible. En este caso, un eje puede ser seleccionado de forma separada de sus nodos origen y destino, ya que el usuario puede operar sobre él de una manera diferente. Puede mover su etiqueta en el modo visible o hacer doble clic sobre él para que la etiqueta aparezca o desaparezca en el modo semivisible.

Por lo tanto, el objeto *ControladorGrafo* se registra como escuchador en todos los grafos que se crean y, de este modo, es capaz de recibir todos los eventos provocados por el cambio de selección cada vez que el usuario haga clic sobre un nodo distinto del grafo.

Finalmente, como ya habrá notado el atento lector, esta clase actúa prácticamente como un "puente" entre los dos grandes bloques en los que se ha dividido la aplicación. Hasta este punto del documento se han descrito los servicios que *ControladorGrafo* ofrece al bloque de la interfaz gráfica. La funcionalidad de esta clase que resta por describir es el pilar central de esta comunicación, por lo que se le ha dedicado el siguiente apartado a comentarla, aunque brevemente, de forma exclusiva.

Con esto, pues, queda terminada la descripción de la interfaz gráfica. Ya se comentó que se trataba de una versión resumida donde se abordaban los aspectos más destacables de su implementación. Existen algunas clases más en este bloque que se ocupan de tareas auxiliares y que no merece la pena comentar aquí en detalle. Una

descripción completa puede consultarse en la documentación de clases y los diagramas contenidos en el Pliego II.

5.2.2 Comunicación entre bloques

La parte verdaderamente destacable de *ControladorGrafo*, al menos en el ámbito de este documento, reside en el tercero de los puntos mencionados durante la enumeración de las funciones principales de esta clase: centralización de las operaciones de intercambio de información con la lógica de negocio.

En la siguiente figura se muestra un diagrama de clases que, a través de las relaciones que contiene, pone de relieve el canal de comunicación existente entre los dos grandes bloques de la aplicación y el papel que desempeña *ControladorGrafo* en él.

Figura 5.7 Clases principales en la comunicación entre bloques en HyperAuthor.

Dentro de este rol, *ControladorGrafo* se encarga de ejecutar las acciones necesarias en cada momento sobre la clase *Encoder*, núcleo de la lógica de negocio, de manera que hace este bloque transparente a ojos de la interfaz gráfica. Estas acciones son básicamente aquellas relacionadas con las tareas de persistencia de grafos, que se expondrán más adelante.

Lo verdaderamente importante a destacar en este apartado es el mecanismo de transferencia entre bloques de los grafos de usuario (se denominará así a aquellas obras hipertextuales desarrolladas en HyperAuthor por los usuarios de la aplicación), implementado también por la clase *ControladorGrafo*. Este aspecto supuso uno de los puntos más delicados en el desarrollo de la aplicación. El problema no es trivial. En primer lugar, las obras de usuario existen de maneras distintas en los distintos bloques: existen en forma de árbol en el bloque de lógica de negocio mientras que tienen forma de grafo en la interfaz gráfica. Son dos estructuras diferentes y, sin embargo, era necesario poder trasladar la información de un bloque a otro para que todo el desarrollo y uso de la herramienta tuviese sentido.

La traslación de grafos de usuario de un bloque a otro, con la conversión en estructura que esto implica, queda resuelta e implementada al completo en dos métodos de *ControladorGrafo*.

- *resumeGraph()*, que genera un array de objetos *HyperGraphCell* que representa al grafo. Cada una de estas celdas almacenadas en el array contiene su propio objeto *Nodo* con la información específica sobre ésta.

- *buildGraph(),* que genera un objeto *HyperGraph* a partir de un array de objetos *HyperGraphCell.* Al igual que antes, de nuevo cada una de estas celdas almacenadas en el array contiene su propio objeto *Nodo* con información específica.

Ya se ha comentado que los objetos *Nodo* actúan como objetos de intercambio entre los dos grandes bloques, por lo que contienen información específica aplicable a cada ámbito. Pues bien, un motivo fundamental para la construcción de la clase *Nodo* fue precisamente el proporcionar una solución al problema del intercambio de los grafos, aunque, como ha quedado expuesto, posteriormente se dotó a esta clase de muchas otras capacidades.

Como es lógico, un grafo con unas características como las de los grafos creados en HyperAuthor, que parten de una secuencia lineal de la que surgen ramificaciones que antes o después vuelven a esta secuencia, puede transformase sin mayor problema en un array que contenga los objetos representativos de sus nodos. Los nodos pueden almacenarse en el array sin ningún orden específico y para recuperar el grafo bastará con localizar el primer nodo (aquél que no tiene ningún nodo "anterior" a él) y recorrer avanzando a partir de él la secuencia lineal principal a la vez que, cada vez que se pasa por un nodo nuevo, se conectan a él todas sus ramificaciones.

Por último, es digno de comentario en este apartado otra clase de la que se puede decir que también "tiene vida" en los dos bloques, aunque en este caso de una forma más sutil. Se comentaba en el apartado anterior que *ControladorGrafo* almacena información de control sobre los grafos en uso. Pues bien, parte de esta información está contenida en objetos de tipo GraphInfo, uno por diagrama en uso.

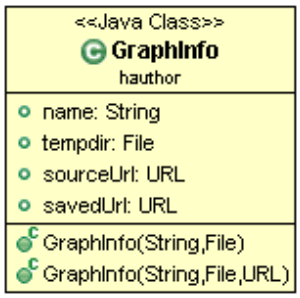


Figura 5.8 Vista UML de la clase *GraphInfo* utilizada en HyperAuthor

Las propiedades de este tipo de objetos hablan por sí mismas, y como puede fácilmente intuirse, están relacionadas con las funciones que debe soportar la lógica de negocio, así que no tiene mucho sentido desarrollarlas aquí. En cualquier caso, se puede

consultar una descripción más detallada de las mimas en la documentación de clases. En este momento no se va a insistir sobre ellas.

5.2.3 La lógica de negocio

El siguiente diagrama de clases contiene una representación resumida de la implementación final de la lógica de negocio. Para una referencia completa, el lector puede consultar el Pliego II: Planos.

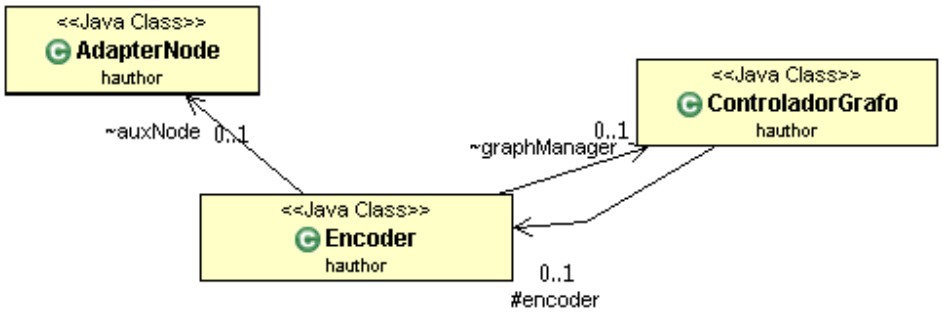


Figura 5.9 Clases principales de la lógica de negocio de HyperAuthor

En el apartado 4.3.4 se describieron las funciones que debía realizar este bloque, entre las que destaca el almacenamiento y organización de los distintos árboles (ya se señaló con anterioridad en el apartado 4.2.2 que se denomina árbol al equivalente de un grafo dentro del bloque de la lógica de negocio) que están en uso en un momento dado, de realizar las modificaciones necesarias sobre los mismos durante la ejecución del programa y de las labores relativas a la persistencia.

Del mismo modo que en el anterior apartado, es importante aclarar antes de empezar que el objetivo de esta descripción no es un listado exhaustivo de los métodos y variables de estas clases, sino una descripción aclaratoria de las mismas y sus puntos esenciales. Del mismo modo, el diagrama que muestra la figura 5.9 es también un diagrama resumido en el que sólo están reflejadas las clases esenciales.

La clase *ControladorGrafo*, descrita en el apartado anterior, se incluye en el diagrama de la figura 5.9 por completitud. Por lo demás, este módulo se compone de una clase central denominada *Encoder* y otra auxiliar, la clase *AdapterNode*.

La clase *Encoder* almacena los árboles correspondientes a los distintos grafos de usuario que se encuentran abiertos en la aplicación, y la más importante de las funciones que lleva a cabo es la persistencia o almacenamiento y recuperación desde ficheros de los hipertextos creados por los usuarios.

Recordemos en este punto lo que ya se ha comentado en varias ocasiones, y que no es más que el hecho de que este bloque de la aplicación implementa los grafos de usuario

mediante árboles DOM. En el apartado 4.2.4 se detallaron las razones que motivaron la adopción de esta tecnología.

Con el fin de almacenar los diagramas en uso, la clase *Encoder* mantiene en memoria una tabla que contiene los objetos *Document* (ver [37]) con los que se representa cada hipertexto creado por el usuario. La tabla está indexada por los objetos *GraphInfo* que mantiene la clase *ControladorGrafo*, de modo que estos objetos sirven de referencia y de intercambio entre ambas clases, identificando a un único árbol.

Es interesante hacer notar que no es necesario que el árbol almacenado en memoria esté ordenado de un modo especial, ya que para reconstruir el grafo a partir de él basta con recorrer uno a uno todos los nodos que lo componen. Cada nodo contiene información sobre su conectividad, por lo que se crearán todas las conexiones necesarias para construir un diagrama válido.

La persistencia de los árboles se implementa de modo automático mediante las clases que proporciona la propia librería JDom. Para guardar en disco un árbol de los que se encuentran abiertos en la aplicación, *Encoder*, en su método *saveGraph()* crea el objeto *Diagram* pertinente a partir del árbol utilizando *DocumentBuilderFactory* [37]. Posteriormente construye un objeto *Transformer* [37] que se encargará, como su propio nombre indica, de transformar el documento que se acaba de crear en un fichero de texto. El fichero se crea en el directorio temporal del grafo correspondiente y toma el nombre por defecto. En último lugar, mediante el método *zipFile()* se accede a este directorio temporal y se comprime su contenido, dando lugar al fichero definitivo con el nombre y en la ruta escogidos por el usuario y que recibe este método como parámetro en forma de url.

Es importante resaltar el hecho de que el objeto *Transformer* se configura de modo que especifique el tipo de documento XML al que pertenecerá el almacenado en el archivo, dicho en otras palabras, se configura para que el archivo resultante referencie en su tag *DOCTYPE* a la DTD[39] que se desarrolló para HyperAuthor.

La DTD define el lenguaje XML en el que estarán escritos los archivos exportados de HyperAuthor que contienen la información sobre la estructura del hipertexto, limitando de esta manera la forma tanto de los árboles DOM en memoria como la de los archivos de texto almacenados en disco. Dicho en otras palabras, esto significa que contiene todas las reglas gramaticales a partir de las cuales pueden estar construidos ambos, tal y como ya se adelantó de forma más exhaustiva en el apartado 4.2.4.

En la recuperación de un hipertexto almacenado en disco, *Encoder* realiza en primer lugar las labores de decomprensión del archivo correspondiente a un fichero temporal creado de manera específica para ese hipertexto. Esta labor se lleva a cabo en el método *unzipFile()* . Una vez descomprimido el archivo original, la herramienta tiene acceso libre al fichero XML en el que se almacena el árbol correspondiente. El método *openGraph()* hace uso con este fin de un objeto *DocumentBuilder* validante [39] que parsea el archivo de texto indicado y comprueba al mismo tiempo que este archivo es un documento xml bien formado, es decir, que es válido conforme a la DTD de referencia creada para HyperAuthor. De este modo se garantiza algo que es evidentemente imprescindible: sólo

se podrán abrir en la herramienta aquellos documentos que representan grafos de usuario válidos.

La función de validación hace referencia en este caso al proceso mediante el cual se comprueba que un árbol existente en el módulo de la lógica de negocio es correcto gramaticalmente. Como se puede deducir del párrafo anterior, la validación se reduce a comprobar que la estructura del árbol DOM cumple con las reglas especificadas en el DTD, y por lo tanto, este proceso también se lleva a cabo mediante las funciones proporcionadas por la librería JDom. En concreto, es el propio objeto *DocumentBuilder* descrito anteriormente el que se encarga de esta tarea.

El objeto *DocumentBuilder* devuelve un *Document* que será almacenado en la tabla correspondiente y, desde ese mismo momento, se encontrará disponible en la aplicación para su edición. La clase *Encoder* en este momento, y partir del documento mencionado, construye el grafo y se lo presenta al usuario en pantalla. El encargado de esta tarea es el método *graphToArray()*, que "lee y procesa" el documento y crea un array de objetos en el que almacena *HyperGraphCells* que contienen su objeto *Nodo* correspondiente. Este array lo recibe la clase *ControladorGrafo* que invoca su método *buildGraph()* para construir y representar el grafo en pantalla.

De nuevo los objetos *Nodo* actúan como objetos de intercambio, siendo la clase receptora de los mismos la responsable de la construcción de un diagrama correcto a partir de ellos. En este punto es donde entra en juego la clase auxiliar *AdapterNode*. Se trata de una clase que envuelve a un nodo DOM y que pretende ocultar de esta manera la complejidad de la operación con los objetos *Node* y *Element* de JDom [37].

Más detalles sobre la implementación de los métodos descritos hasta este momento, así como sobre la utilización de los métodos proporcionados por las clases de JDom en su implementación pueden ser consultados en la documentación de clases [37].

A modo de una descripción genérica de la clase *Encoder* se puede decir que se implementó como un repositorio de utilidades de especial relevancia en las tareas de persistencia. En efecto, *Encoder* implementa funciones de gran relevancia en el conjunto de la aplicación, pero lo hace proporcionando un conjunto de métodos a modo de funciones inconexas que serán invocadas de manera mayoritaria desde las clases *HyperAuthor* y *ControladorGrafo* (en su calidad de intermediaria con la interfaz gráfica).

En este sentido, es interesante hacer hincapié en el modo en el que los árboles se crean y se eliminan en la clase *Encoder*. Un aspecto muy importante es que, cuando un usuario crea un diagrama en la interfaz gráfica, éste sólo se almacenará en forma de árbol en la clase *Encoder* en el momento en el que se invoque una acción de guardado. Esta acción, debido a que implica acciones relacionadas con la persistencia, pertenece a la lógica de negocio a pesar de que se encuentre accesible desde la interfaz. Por supuesto, la interfaz tan solo ordena la acción al objeto *ControladorGrafo* que de nuevo actúa como mediador y, a su vez, invoca los métodos correspondientes de *Encoder*.

5.3 HyperViewer

Hasta aquí se han descrito con detalle los aspectos más importantes de la fase de implementación de la herramienta creadora, HyperAuthor, justificando las decisiones adoptadas durante la misma. Por lo tanto, en lo que a la fase de implementación se refiere, resta sólo hacer lo propio con la herramienta de visualización: HyperViewer.

5.3.1 La interfaz gráfica

En el siguiente diagrama UML se muestran las clases más significativas en la estructura software del módulo de interfaz gráfica. Para una referencia completa, el lector puede consultar el Pliego II de este documento: Planos.

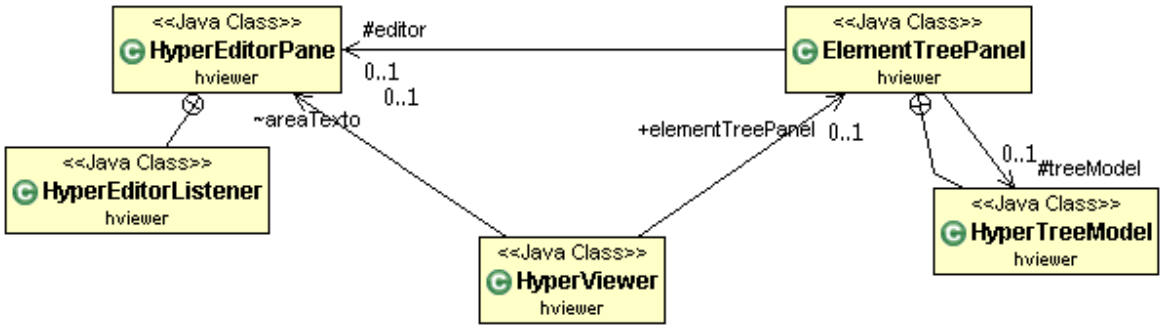


Figura 5.10 Clases principales de la interfaz gráfica de HyperViewer.

Toda la funcionalidad desarrollada para esta herramienta se encuentra dentro del paquete **hviewer**. La clase central del paquete (y de la aplicación completa) es *HyperViewer*. Esta clase contiene el método *main* y se encarga de crear y mostrar la interfaz gráfica en pantalla así como de gestionar los flujos de interacción con el usuario proporcionando:

- Los controles que el usuario puede accionar (botones, menús, teclas de acceso rápido, menús emergentes…).
- El soporte necesario para otros objetos que necesitan residir en un contenedor gráfico.
- Los mecanismos de realimentación hacia el usuario (por ejemplo, mensajes de error).

En último lugar, aunque no por ello menos importante, lleva a cabo la tarea de intermediación en la comunicación entre el módulo de la interfaz gráfica y la lógica de negocio, una tarea fundamental para el buen funcionamiento de la aplicación.

En resumen, se podría decir que *HyperViewer* implementa lo que se ha denominado anteriormente "interfaz de usuario" (ver Figura 4.6), que se ajusta a lo descrito en el apartado 4.3.3.

Es importante hacer notar que en esta herramienta no es necesario el uso de la biblioteca JGraph, ya que no se va a llevar a cabo la representación gráfica de la estructura hipertextual que tenía lugar en la herramienta creadora. HyperViewer, por lo tanto, no tiene conocimiento de la estructura completa de la obra. Para el visor la información relevante es la perteneciente al plano de contenido. En este plano se encuentran contenidos los enlaces, de forma que la herramienta puede conocer, al menos de forma local y aislada, las conexiones entre un determinado nodo y sus adyacentes.

Por este motivo, y del mismo modo que ocurría en HyperAuthor, los nodos deben actuar como contenedores de toda aquella información relativa al plano de contenido y de la cual es necesario disponer, ya que ésta será la que se muestre en pantalla. Precisamente para suplir esta necesidad se creó, a semejanza de la desarrollada durante la implementación de la herramienta creadora, la clase *Nodo*. Los detalles de la misma difieren un poco de los de su predecesora, ya que ha sido adaptada a las necesidades concretas de esta herramienta. La descripción se encuentra en la siguiente figura.

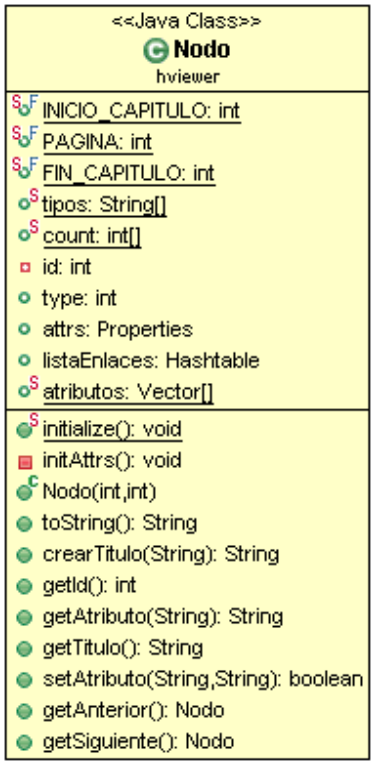


Figura 5.11 Vista UML de la clase Nodo empleada en HyperViewer.

Este objeto *Nodo* se emplea como soporte para almacenar toda aquella información necesaria del mismo perteneciente al plano de contenido, así como alguna información relativa a las conexiones existentes. Estos objetos *Nodo* son creados cuando la herramienta lee un hipertexto almacenado en disco y se almacena en memoria en un array de objetos perteneciente a la clase Datos, contenida en el módulo de la lógica de negocio y que se explicará más adelante. Son también los objetos que se almacenarán en los nodos del árbol

con el que se crea el histórico de nodos visitados, en la clase *ElementTreePanel* cuya explicación se llevará a cabo también en los párrafos posteriores.

Es también importante mencionar que los objetos *Nodo* se emplean como objetos de intercambio con las clases pertenecientes a la lógica de negocio. Esta relación se verá más adelante en el punto 5.3.2.

Como se puede intuir a partir del diagrama de la figura 5.11, la clase *Nodo* lleva gran parte del peso de gestión de los nodos y sus conexiones. En concreto:

- Cada instancia de clase contiene atributos como el identificador, el título o el texto que permiten distinguir unos nodos de otros. Estos atributos son generados automáticamente por el constructor de la clase, que siempre recibirá como parámetros el identificador de nodo (que debe ser único dentro de cada grafo) y el tipo que debe tener la instancia. Se iniciarán con sus valores por defecto, que deberán ser modificados por la herramienta durante el momento de construcción de la obra antes de la presentación por pantalla de la misma al usuario.

- Mantiene en una tabla la información relativa a los enlaces contenidos en un determinado nodo. En esta tabla se almacenan tanto los nodos anterior y siguiente al nodo actual en la secuencia lineal a la que pertenece, así como el primer nodo de cada uno de los bucles que tienen como origen a ese nodo en concreto. De esta manera la herramienta, que no conoce la completitud de la estructura hipertextual, puede moverse por ella mediante el conocimiento del "siguiente salto" que adquiere a partir de la información sobre su propia conectividad almacenada en la instancia concreta de la clase *Nodo*.

De un modo más general, se puede decir que esta clase *Nodo* es la portadora de toda la información a representar en pantalla, por lo que constituye el pilar fundamental de la herramienta. No en vano, tiene presencia tanto en el ámbito de la interfaz gráfica como en el de la lógica de negocio, que se estudiará en el siguiente apartado, y actúa incluso como soporte para el intercambio de información entre ambos módulos.

Por otro lado, y siguiendo los preceptos establecidos en 4.3.3, HyperViewer pretende conseguir un área de lectura lo más adecuada posible, por lo que uno de los objetivos básicos era reducir al mínimo el número de menús, herramientas y demás formas de interacción con HyperViewer que desviaran la atención del usuario del propio relato para ponerla en la herramienta.

Por este motivo, todas las acciones que el usuario puede realizar en cada momento de la ejecución se encuentran "integradas" dentro del propio área de lectura. Adicionalmente, en el panel izquierdo de la aplicación se ha incluido aquello que se ha estado denominando a lo largo de este documento como ayuda a la navegación. El panel "Mi historia" hace las veces de histórico de nodos visitados, reconstruyendo de esta manera el camino de lectura que el propio lector ha escogido de entre todos los posibles y

facilitando, además, el acceso rápido a cualquiera de estos nodos. Las clases *HyperEditorPane* y *ElementTreePanel* surgen para hacer posible esta minimización de elementos ajenos a la lectura presentes en pantalla durante la misma.

En primer lugar, para mostrar el contenido de los nodos en pantalla se decidió utilizar el mismo tipo de panel que ya se utilizaba en la herramienta de autor, reutilizando de esta manera gran parte del código desarrollado. Sin embargo, este panel debía ofrecer en este caso una funcionalidad adicional: ser capaz de gestionar los eventos de ratón generados por el usuario en su interacción con la pantalla para navegar por el hipertexto. Con esta finalidad se implementa *HyperEditorPane*. Esta clase, al igual que el panel utilizado en HyperAuthor, extiende de *JEditorPane* [31], una clase proporcionada por Swing que puede ser configurada de forma que se le dote de la capacidad de entender y procesar código HTML. Es importante recordar en este punto que el contenido de los nodos se almacena con formato HTML y, por lo tanto, un editor con capacidad para procesar este tipo de lenguaje se revela de especial utilidad para la aplicación.

HyperEditorPane tiene asociados un *HypeLlinkListener* [31] y un *HyperEditorListener*. El *HyperLinkListener* es una clase proporcionada por Swing que permite captar los eventos generados por el usuario cuando hace clic sobre un determinado link. En el caso de HyperViewer, cuando se genera un evento de este tipo se invoca el método *followLink()* contenido en la clase *HyperEditorPane.* En este método, se busca el nodo destino del enlace activado y se muestra su contenido por pantalla usando el método *display(),* también de *HyperEditorPane.*

Por otro lado, *HyperEditorListener*, el segundo de los objetos asociados a *HyperEditorPane,* es una clase que extiende de *MouseAdapter* [31]. Esta es una clase desarrollada "a medida" como clase interna de *HyperEditorPane* y que permite crear tres zonas diferenciadas a nivel lógico en el área de lectura. Cuando el usuario hace clic en la pantalla, *HyperEditorListener* recibe el evento correspondiente y, basándose en las coordenadas del evento, es capaz de discernir sobre cuál de las zonas se ha producido. Si se trata de las zonas izquierda o derecha, el nodo anterior o siguiente al actual se muestra en pantalla usando de nuevo el método *display()* de *HyperEditorPane.* Una pequeña mención en este momento es necesaria para aclarar que, para acceder al nodo anterior o siguiente al actual, se utilizan los métodos correspondientes de la clase *Nodo*.

Note el lector en este momento que el método *display()* se encarga únicamente de la presentación del nodo en pantalla, pero no de la correspondiente actualización del histórico. Esta actualización del histórico de nodos se lleva a cabo de manera inmediatamente posterior mediante la invocación de métodos contenidos en *ElementTreePanel*.

El motivo de la separación de las tareas de visualización de nodos y actualización del histórico se debe a que no siempre la visualización de un nodo debe dar lugar a su inclusión en el camino de lectura seguido por el usuario. Este es el caso de cuando la presentación de un nodo en pantalla tiene lugar como consecuencia de que el usuario haya seleccionado el nodo correspondiente en el panel que contiene el histórico de nodos visitados. En ese caso, el nodo debe ser visualizado en pantalla pero no incluido en el

historial, ya que éste debe representar un trayecto de lectura que pueda ser seguido por el lector si en algún momento desea volver sobre sus pasos para releer algunos fragmentos del relato. Como ya se ha comentado, el proceso de actualización tiene relación con la clase *ElementTreePanel,* por lo que se explicará más adelante cuando se profundice en los detalles de esta clase.

Después de esta aclaración, retomemos el hilo del discurso. Si por el contrario es en la zona central del área de lectura en la que el usuario ha hecho clic, la respuesta de la herramienta a este evento debe ser ocultar o mostrar los enlaces contenidos en el nodo, o, en su defecto, resaltar las zonas anterior y siguiente si el nodo en cuestión no contiene ningún enlace. Con este propósito *HyperEditorListener* invoca a los métodos *activarEnlaces()* o *desactivarEnlaces()* de la clase *HyperEditorPane.* Es interesante comentar que la "iluminación" de las zonas anterior o siguiente se lleva a cabo mediante la inclusión de una imagen semi-transparente del tamaño de estas zonas. Estas imágenes están siempre presentes en el área de trabajo, pero sólo se hacen visibles si el panel no es opaco. Por lo tanto, para mostrar y ocultar estas imágenes que proporcionan el efecto de resaltado, basta con jugar con la opacidad del panel.

Un último apunte sobre la clase *HyperEditorPane*. Cuando un usuario hace clic sobre un enlace, tanto *HyperEditorListener* como *HyperLinkListener* reciben el evento correspondiente. Sin embargo, el único destinatario de un evento generado de este modo debería ser *HyperLinkListener*, por lo que cuando ambos escuchadores recibían el evento la herramienta mostraba un comportamiento inadecuado. Para solucionar esta situación se creó la variable clickOnEnlace. Esta variable toma el valor true cada vez que *HyperLinkListener* recibe un evento, de manera que cuando el evento paralelo sea recibido por *HyperEditorListener,* éste realiza una comprobación sobre esta variable, descartando el evento si su valor es true y cambiándolo en ese momento a false para poder recibir eventos futuros. Si cuando el evento se recibe clickOnEnlace tiene el valor false, el evento en *HyperEditorListener* se gestiona de manera ordinaria sin el menor problema.

Como ya se ha comentado brevemente, la clase *ElementTreePanel* ayuda a *HyperEditorPane* en la labor de ocultar al usuario, en la medida de lo posible, los menús y barras de herramientas tan habituales en otro tipo de aplicaciones. Esta clase hereda de *JPanel* [31] y contiene como elemento principal un objeto *JTree* [40], que proporciona una vista similar al de muchas ventanas de ayuda de herramientas comerciales. La elección de este tipo de objeto se llevó a cabo principalmente por motivos de escalabilidad. Hasta el momento, el único mecanismo de ayuda a la navegación implementado es la función "Mi historia", pero la herramienta lectora deberá incorporar en el futuro muchas más funcionalidades. El elemento *JTree* permite utilizar el mismo panel izquierdo para desplegar y ocultar las funcionalidades deseadas en cada momento por parte del usuario, de manera que el espacio dedicado a las mimas, por mucho que estas incrementen en número, será siempre el mismo.

Es la propia clase *ElementTreePanel* la que contiene los métodos que permiten gestionar y controlar el objeto *JTree* a alto nivel, ocultando de esta manera toda la complejidad de clases y métodos necesarios para llevar a cabo esta tarea. Como en casi todos los componentes Swing [31], existen dos clases importantes implicadas, la vista y el

modelo, siendo éste último el responsable del control y actualización de la primera. *JTree* representa la vista, y cualquier clase que implemente la interfaz *TreeModel* puede actuar como modelo. *HyperTreeModel* es una implementación concreta del modelo que proporciona Swing por defecto, *DefaultTreeModel* [31]. *HyperTreeModel* se ha desarrollado como una clase interna de *ElementTreePanel* con la finalidad de poder añadir funcionalidades adicionales al modelo del árbol, y es la encargada de contener los datos que se visualizarán en pantalla como componentes del árbol. Sin embargo, es necesario hacer notar que la implementación por defecto hubiera sido suficiente para cubrir las necesidades actuales de la herramienta.

Por este mismo motivo, la clase *ElementTreePanel* implementa la interfaz *TreeSelectionListener,* de manera que puede responder ante cambios en la selección de nodos dentro del árbol. En consecuencia, los métodos más importante de la clase *ElementTreePanel* son los métodos *updateMyHistory()* y *valueChanged(),* este último de obligada creación como consecuencia de implementar *TreeSelectionListener*.

El método *updateMyHistory()* es el responsable de la actualización del histórico de nodos. Con este fin se emplea la clase *DefaultMutableTreeNode,* que representa los nodos que van a ser añadidos al árbol (el histórico). Esta clase cumple con el requisito necesario de implementar la interfaz *TreeNode* [31] para poder ser admitido como dato válido por *HyperTreeModel* y además actúa como contenedor de objetos de información. En el método *updateMyHistory()* se crea un objeto *DefaultMutableTreeNode* [31] al que se asocia, mediante el uso de su método *setUserObjetct(),* el objeto *Nodo* correspondiente. Una vez creado y con contenido, el objeto *DefaultMutableTreeNode* es añadido al modelo del árbol invocando el método *insertNodeInto()*, lo que provoca la actualización de la vista en la que aparece el nodo correspondiente.

Por otro lado está el método *valueChanged()*. Este método se invoca de forma automática cada vez que la selección en el histórico cambia y se ocupa de mostrar en pantalla el contenido del nodo seleccionado. En concreto, el proceso se lleva a cabo recuperando el objeto *DefaultMutableTreeNode* seleccionado, del que se extrae el objeto de información que contiene mediante el método *getUserObject().* Una vez se ha recuperado el *Nodo*¸ se invoca el método *display()* de *HyperEditorPane*, sin llevar a cabo en este caso la actualización del propio histórico.

Con esto, pues, queda terminada la descripción de la interfaz gráfica. Ya se comentó que se trataba de una versión resumida donde se abordaban los aspectos más destacables de su implementación. Existen algunas clases más en este bloque que se ocupan de tareas auxiliares y que no merece la pena comentar aquí en detalle. Una descripción completa puede consultarse en los diagramas contenidos en el Pliego II de esta memoria.

5.3.2 Comunicación entre bloques

Este apartado constituye fundamentalmente una repetición de lo ya expuesto y detallado en el apartado homónimo perteneciente a la implementación de HyperAuthor. Esto es debido a que si estructuralmente las dos herramientas son muy similares, basadas en la misma filosofía y la misma arquitectura, es razonable la similitud también en cuanto a la forma de comunicación entre los dos grandes bloques que las componen.

A este respecto, la parte verdaderamente destacable de *HyperViewer*, al menos en el ámbito de este documento, reside en el último de los puntos mencionados durante la enumeración de las funciones principales de esta clase: centralización de las operaciones de intercambio de información con la lógica de negocio.

En la siguiente figura se muestra un diagrama de clases que, a través de las relaciones que contiene, pone de relieve el canal de comunicación existente entre los dos grandes bloques de la aplicación y el papel que desempeña *HyperViewer* en él.

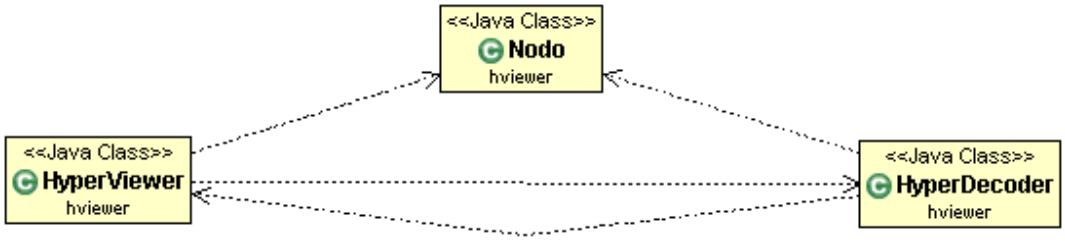


Figura 5.12 Clases implicadas en la comunicación entre bloques en HyperViewer.

Dentro de este rol, *HyperViewer* se encarga de ejecutar las acciones necesarias en cada momento sobre la clase *HyperDecoder*, clase que actúa como núcleo de la lógica de negocio, de manera que hace este bloque transparente a ojos de la interfaz gráfica. Estas acciones son básicamente aquellas relacionadas con las tareas de persistencia de hipertextos, que se expondrán más adelante.

Lo verdaderamente importante a destacar en este apartado es el mecanismo de transferencia entre bloques de los hipertextos. Este aspecto no resultó tan delicado como en el caso de HyperAuthor, ya que el problema de traducción de estructuras ya había sido resuelto durante la implementación de la herramienta creadora.

A modo de recordatorio es necesario hacer notar que el problema no es trivial. En primer lugar, las obras de usuario existen de maneras distintas en los distintos bloques: existen en forma de árbol en el bloque de lógica de negocio mientras que tienen forma de array (que contiene objetos *Nodo*) en la interfaz gráfica. Son dos estructuras diferentes y,

sin embargo, era necesario poder trasladar la información de un bloque a otro para que todo el desarrollo y uso de la herramienta tuviese sentido.

En este caso, el flujo de información entre un bloque y otro es unidireccional: la aplicación debe ser capaz de convertir el árbol DOM en un array de objetos *Nodo* pero la conversión en el sentido contrario no es necesaria. Por lo tanto, la estructura en forma de árbol pierde protagonismo ya que la aplicación trabaja de forma mayoritaria con el array de nodos. Los objetos *Nodo* se convierten por lo tanto en la base del intercambio de información entre los dos grandes bloques de la aplicación. Basta con facilitar el acceso a estos objetos desde ambos bloques para que la comunicación entre ambos pueda llevarse a cabo de forma efectiva.

Esta clase debe contener por lo tanto, como ya ocurría en HyperAuthor y obedeciendo a las mismas razones, información específica aplicable a cada uno de los dos ámbitos. Facilitar el intercambio de información entre los dos bloques de los que se compone estructuralmente la aplicación fue el motivo fundamental para la construcción de la clase *Nodo,* aunque posteriormente se ha dotado a esta clase de muchas otras capacidades, entre ellas la posibilidad de acceder a los nodos directamente conectados con un nodo en concreto.

Como se puede deducir de los párrafos anteriores, en HyperViewer un grafo va a estar representado por un array de objetos *Nodo.* Ya se comentó en el apartado correspondiente de la herramienta creadora (ver 5.2.2) que un grafo con unas características como las de los grafos creados en HyperAuthor, que parte de una secuencia lineal de la que surgen ramificaciones que antes o después vuelven a esta secuencia, puede transformase sin mayor problema en un array que contenga los objetos representativos de sus nodos. Los nodos pueden almacenarse en el array sin ningún orden específico y para recuperar el grafo bastará con localizar el primer nodo (aquél que no tiene ningún nodo "anterior" a él) y recorrer avanzando a partir de él la secuencia lineal principal a la vez que, cada vez que se pasa por un nodo nuevo, se conectan a él todas sus ramificaciones.

Por último, es digno de comentario en este apartado que para facilitar el acceso a este array que representa el grafo en uso (note el lector que en HyperViewer, a diferencia de lo que sucedía en HyperAuthor, únicamente se permite la apertura de un único grafo de manera simultánea en la aplicación), se creó la clase *Datos,* cuya misión se detallará más adelante.

5.3.3 La lógica de negocio

El siguiente diagrama de clases contiene una representación resumida de la implementación final de la lógica de negocio. Para una referencia completa, el lector puede consultar el Pliego II: Planos.

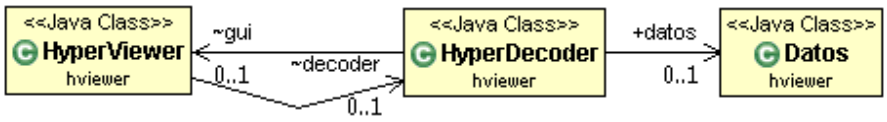


Figura 5.13 Estructura resumida de la lógica de negocio de HyperViewer.

En el apartado 4.2.4 se describieron las funciones que debía realizar este bloque, entre las que destaca el almacenamiento y gestión del hipertexto en uso.

Del mismo modo que en el anterior apartado, es importante aclarar antes de empezar que el objetivo de esta descripción no es un listado exhaustivo de los métodos y variables de estas clases, sino una descripción aclaratoria de las mismas y sus puntos esenciales. De igual forma, el diagrama que muestra la figura 5.13 es también un diagrama resumido en el que sólo están reflejadas las clases esenciales.

La clase *HyperViewer*, descrita en el apartado anterior, se incluye en el diagrama de la figura 5.13 por completitud. Por lo demás, este módulo se compone de una clase central denominada *HyperDecoder* y otra auxiliar, la clase *Datos*.

La clase *HyperDecoder* tiene como función fundamental el llevar a cabo las funciones relacionadas con la persistencia. En este caso, la única tarea a realizar a este respecto consiste en la recuperación desde fichero de los hipertextos creados por los usuarios; no ha sido necesario la implementación de la tarea de almacenamiento ya que, al menos hasta el momento, la herramienta no permite la edición, sino únicamente la visualización de las obras hipertextuales.

Recordemos en este punto lo que ya se ha comentado en varias ocasiones, y que no es más que el hecho de que este bloque de la aplicación implementa los grafos de usuario mediante árboles DOM. En el apartado 4.3.4 se detallaron las razones que motivaron la adopción de esta tecnología.

La persistencia de los árboles se implementa de modo automático mediante las clases que proporciona la propia librería JDom. Antes de describir el proceso llevado a cabo en la recuperación de un hipertexto almacenado en disco, es necesario recordar la forma en que se exportan los documentos desde la herramienta creadora.

La estructura del hipertexto se almacena en un fichero XML que contiene, además, la información relativa al plano de contenido. Para dar soporte a la inserción de imágenes en el texto así como soportar el texto con formato, junto con el fichero XML se crea una CSS que contiene los estilos creados por el usuario y una carpeta que contiene las imágenes que el usuario insertó en el texto. Todo esto se guarda en una carpeta que se comprime usando formato zip. El fichero comprimido resultante tendrá extensión .htxt y el nombre que el usuario haya elegido. Para una descripción más detallada del proceso de exportación llevado a cabo en HyperAuthor el lector puede dirigirse al apartado 4.2.2 de este documento.

En la recuperación de un hipertexto almacenado en disco *HyperDecoder* realiza, en primer lugar, las labores de decomprensión del archivo correspondiente a un fichero temporal creado de manera específica para ese hipertexto. Esta labor se lleva a cabo en el método *unzipFile()*. Una vez descomprimido el archivo original, la herramienta tiene acceso libre al fichero XML en el que se almacena el árbol correspondiente. El método *openGraph()* hace uso con este fin de un objeto *DocumentBuilder* validante [39] que parsea el archivo de texto indicado y comprueba al mismo tiempo que este archivo es un documento xml bien formado, es decir, que es válido conforme a la DTD de referencia creada para HyperAuthor y que se utiliza también en esta aplicación. De este modo se garantiza algo que es evidentemente imprescindible: sólo se podrán abrir en la herramienta aquellos documentos que representan grafos de usuario válidos.

Ya se explicó en un apartado análogo a éste relativo a la lógica de negocio en HyperAuthor la función de la DTD. Ésta no es otra que la de contener las reglas gramaticales que deben cumplir los ficheros exportados de HyperAuthor y que contienen toda la información sobre el hipertexto creado utilizando esta herramienta. HyperViewer necesita conocer el contenido de esta DTD para validar estos archivos, ser capaz de comprender su contenido y mostrarlo por pantalla.

La función de validación hace referencia en este caso al proceso mediante el cual se comprueba que un árbol existente en el módulo de la lógica de negocio es correcto gramaticalmente. Como se puede deducir del párrafo anterior, la validación se reduce a comprobar que la estructura del árbol DOM cumple con las reglas especificadas en la DTD, y por lo tanto, este proceso también se lleva a cabo mediante las funciones proporcionadas por la librería JDom. En concreto, es el propio objeto *DocumentBuilder* descrito anteriormente el que se encarga de esta tarea.

El objeto *DocumentBuilder* devuelve un *Document* que contiene ya el hipertexto en forma de árbol DOM por lo que, desde ese mismo momento, se encontrará disponible en la aplicación. La clase *HyperDecoder* en este momento, y partir del documento mencionado, construye un array de objetos *Nodo* que representa al hipertexto en el bloque de la interfaz gráfica. El encargado de esta tarea es el método *graphToArray()*, que "lee y procesa" el documento y crea un array de objetos en el que almacena *Nodo.*

En este punto es donde entra en juego la clase auxiliar *Datos.* Se trata de una clase cuya misión principal es la de ser contenedora del array de nodos que representa al hipertexto en uso y facilitar el acceso al mismo desde cualquier parte de la aplicación, ya que todos los métodos alojados en la clase *Datos* pueden ser accedidos de forma estática. Proporciona también diversos métodos para la recuperación de nodos en base a determinadas condiciones, por ejemplo, el método *getNodoByID()* devuelve el nodo de los contenidos en el array que tiene un id determinado. Este método resulta de especial utilidad para la recuperación del destino de un determinado enlace. De esta forma se pretende ocultar parte de la complejidad y farragosidad de la manipulación directa del array de cara al resto de la aplicación.

Más detalles sobre la implementación de los métodos descritos hasta este momento, así como sobre la utilización de los métodos proporcionados por las clases de JDom en su implementación pueden ser consultados en la documentación de clases.

A modo de una descripción genérica de la clase *HyperDecoder* se puede decir que se implementó como un repositorio de utilidades de especial relevancia en las tareas de persistencia. En efecto, *HyperDecoder* implementa funciones de gran relevancia en el conjunto de la aplicación, pero lo hace proporcionando un conjunto de métodos a modo de funciones inconexas que serán invocadas de manera mayoritaria desde la clase *HyperViewer* (en su calidad de intermediaria con la interfaz gráfica).

5.4 Pruebas

En los apartados anteriores se ha descrito la implementación de la que ha sido, a día de hoy, la versión definitiva de HyperAuthor y su herramienta complementaria, HyperViewer. Sin embargo, de forma previa a esta versión definitiva, se había generado una versión provisional que fue sometida a distintas pruebas.

Como resultado de las pruebas llevadas a cabo, ciertos cambios fueron introducidos en la aplicación. A pesar de que la mayoría de estos cambios se han ido señalando de manera sutil a lo largo del documento, este apartado pretende servir de aglutinación de todos ellos. Por lo tanto, se va a describir con brevedad en qué consistieron los dos procesos de pruebas que se llevaron a cabo sobre la aplicación así como los resultados obtenidos a partir de los mismos.

Los párrafos siguientes no contienen la descripción de un plan de pruebas exhaustivo por la razón fundamental de que no existió como tal. Las pruebas funcionales y de integración básicas ya se habían realizado durante la implementación de la aplicación. Lo que contienen los párrafos siguientes consiste básicamente en una especie de batería de pruebas de estrés sobre la aplicación, realizada en dos fases diferentes y de modos muy distintos.

En una primera fase de pruebas, la aplicación se ofreció a dos voluntarios para que la utilizasen a su voluntad. El objetivo fundamental de esta primera fase era simple, intentar llevar al programa al límite de su funcionalidad para descubrir posibles errores o bugs que pudiera contener. Por lo tanto, esta primera fase de pruebas se centraba de forma exclusiva en la robustez de la aplicación y no tanto en su funcionalidad.

El desarrollo de esta fase requiere el planteamiento de situaciones de considerable complejidad, y durante el desarrollo del mismo la herramienta siguió un proceso de depuración que eliminó los pequeños errores de funcionalidad que aparecían en situaciones extremas de uso.

Una vez la herramienta se vio libre de errores, se ofreció en una segunda fase de pruebas a dos usuarios diferentes con el fin, en esta ocasión, de probar su funcionalidad y usabilidad. Los usuarios debían utilizar la herramienta con el fin para el que fue

concebida, escribir literatura hipertextual, centrándose en proporcionar sugerencias sobre cualquier aspecto de la interfaz o la dinámica de la aplicación.

Pues bien, durante esta fase de las pruebas los usuarios de la aplicación proporcionaron una serie de sugerencias muy valiosas para el uso de la herramienta y que, por lo tanto, se decidieron incluir en su versión definitiva.

- Se añadió a la interfaz un botón para poder "cerrar" la secuencia principal de la estructura enlazando el último nodo con el primero. De esta forma la secuencia principal puede adoptar la forma de un bucle abierto.

- Se incluyó en el editor de nodos un segundo panel de pestañas que permitía la visualización y edición de varios nodos de manera simultánea. Este panel puede activarse y desactivarse a elección del usuario desde las opciones de la herramienta. De esta manera se le ofrece al usuario la opción de personalizar la herramienta para hacer que esta se ajuste lo más posible a sus gustos y forma de trabajar.

- Siguiendo esta filosofía de permitir al usuario personalizar en la medida de lo posible la herramienta, se incluyó también una opción en el menú "Preferencias" que permite modificar el modo de visualización de las etiquetas correspondientes a los ejes en el grafo. Hasta ahora las herramientas permanecían siempre visibles, lo que a veces podía sobrecargar la vista del grafo en pantalla si se estaba trabajando con hipertextos de cierta complejidad. Se incluyeron tres posibles modos de visualización: uno en el que las etiquetas están siempre visibles, otro en el que permanecen siempre ocultas, y un tercero en el que el usuario puede hacer aparecer o desaparecer la etiqueta de un enlace en concreto sin más que hacer doble clic sobre él.

- De igual manera, para facilitar el trabajo con hipertextos complejos, se incluyó lo que comúnmente se conoce como *"tootips"*. El título de los nodos que se muestran en su interior no siempre tienen espacio suficiente para ser visualizados de forma completa. Cuando esto ocurre, se muestran sólo las primeras palabras, lo que podía llevar a situaciones confusas con nodos con títulos parecidos. Al incluir los "*tooltips*", el título completo del nodo se muestra en una etiqueta emergente cuando el usuario deja el ratón sobre el nodo durante unos segundos.

- En último lugar, se modificó la forma en el que se presentaba al usuario la acción de cortar. Hasta este momento, al cortar los nodos desaparecían del área de trabajo cuando el usuario invocaba una acción de cortar sobre ellos, y si no se pegaban o el usuario realizaba otra acción previa al pegado, los nodos se perdían. Para evitar un uso inadecuado de esta función que pudiera desembocar en una mal percepción por parte del usuario de la funcionalidad de la herramienta (algo que sin duda sucedería si el usuario pierde unos nodos que simplemente quería mover de sitio), se cambió este procedimiento. Al cortar unos nodos, éstos no desaparecen por completo del área de trabajo sino

que se marcan como cortados, apareciendo como si estuvieran desactivados. Una vez el usuario invoca una acción de pegado, los nodos desaparecen de su sitio original y se insertan en su nueva posición. De esta manera, si se ejecuta una acción antes del pegado, los nodos no se pierden por completo sino que simplemente dejan de aparecer como cortados. Con el fin de proporcionar un comportamiento adecuado, tampoco puede ejecutarse ninguna acción sobre los nodos cortados mientras estos estén marcados como tal.

En cuanto a las opiniones recavadas, al margen de las sugerencias mencionadas, sobre la utilidad de la aplicación, fueron bastante favorables. A pesar de las limitaciones impuestas a la estructura del hipertexto, no encontraron limitaciones en cuanto al desarrollo de su creatividad aplicada a la obra y todos expresaron su sensación de poder sacar mucho provecho de ella tras una primera fase de familiarización a la herramienta.

Lamentablemente no disponemos de datos más específicos y estructurados sobre estas opiniones, puesto que para eso habríamos tenido que dedicar tiempo a preparar cuestionarios específicos y su posterior análisis, lo que estaba fuera del propósito de las pruebas. Sin embargo, sí se ha querido reflejar en este documento al menos la sensación que la herramienta provocó en sus primeros usuarios.

Capítulo 6:
Conclusiones y trabajos futuros

6.1 Conclusiones

A lo largo de este proyecto se ha pretendido dar una visión clara sobre los diferentes aspectos que han hecho posible el hipertexto, y con ello el nacimiento de un nuevo género literario: la narrativa hipertextual. De esta forma, se han estudiado tanto los aspectos tecnológicos que lo han hecho posible como los aspectos culturales que enmarcan su desarrollo y evolución, haciendo un recorrido por la historia y destacando los principales hitos, eventos, organismos y personas de diversos campos que han contribuido a su puesta en marcha.

Sin embargo, aún queda un largo camino por recorrer hasta que la hiperficción pueda ser considerada como un tipo de literatura madura. Actualmente los que abogan por este tipo de literatura están haciendo un esfuerzo por encontrar un equilibrio entre literatura tradicional e hipertextual de forma que el resultado pueda ser asimilado por el lector. Hoy en día, adoptar un "estilo de vida digital", sacando todo el partido posible de las características que el hipertexto es capaz de ofrecer para crear literatura radicalmente diferente a aquella a la que el gran público está acostumbrado puede suponer pedirle demasiado al lector, que aún es extraño a este nuevo género. Hay que evitar el rechazo por la hiperficción creando obras que aporten algunas novedades que resulten atractivas pero que no pongan en compromiso aquello que se ha logrado durante tantos años de literatura impresa. Por este motivo, el avance y desarrollo en estos aspectos debe ser lento, creando obras que representen únicamente un pequeño paso adelante desde la literatura tradicional y que permita tanto a autores como a lectores descubrir, poco a poco, el lenguaje de esta nueva, a la vez que potente, forma de expresión.

Como tarea fundamental, en este proyecto se han desarrollado dos herramientas de forma paralela que pretenden hacer llegar al público este ya tan comentado nuevo tipo de literatura, la literatura hipertextual. La primera de las herramientas, HyperAuthor, es una herramienta creadora pensada para crear hiperficción que, manteniendo todos aquellos aspectos fundamentales de la literatura tal y como se ha entendido hasta la actualidad, incorpore al mismo tiempo aquellas nuevas características que ofrece el hipertexto y que son capaces de aumentar la unión íntima que debe producirse entre el lector y la obra en una lectura de calidad. En este sentido HyperAuthor se ve obligado a restringir las posibilidades tecnológicas del escritor, adoptando, sin embargo, el firme compromiso de que su creatividad no se vea afectada.

Para reforzar estas características del hipertexto e incorporar algunas otras con la misma finalidad, pero que no tenían sentido en una herramienta de autor, se desarrolla como complemento HyperViewer. Esta es una herramienta de visualización enfocada al lector de hipertexto y cuya misión principal consiste en ofrecer al usuario un entorno de lectura en pantalla lo más adecuado e inmersivo posible. Ofrece también los mecanismos de navegación necesarios para que el lector adopte de una forma natural esta nueva literatura como propia, un tipo nuevo de literatura con una estructura de narración que deja de ser lineal.

El trabajo de desarrollo de estas dos herramientas representa un intento de abordar la creación de hiperficciones nuevas, atrayentes y excitantes sin distanciarse demasiado de

los logros y convenciones alcanzados por la literatura tradicional. Este tipo literatura ha alcanzado desde sus orígenes hasta la actualidad un estado envidiable de madurez y estabilidad. La hiperficción es un género nuevo pero extremadamente prometedor. Lo que ha impulsado este proyecto desde el inicio, y que constituye por lo tanto la columna vertebral del mismo, es la idea de que una combinación adecuada de lo mejor de los dos mundos debería ser capaz de establecer un camino sólido y asentado para conseguir el desarrollo de un nuevo concepto de narrativa.

Este tipo de trabajos que, aún sin sacar todo el partido posible del hipertexto, aprovechan su potencial para empezar a crear un nuevo tipo de literatura son positivos en dos sentidos diferentes: por un lado, incrementan el número de lectores que tienen acceso a este nuevo medio de manera que los lectores se habitúan cuanto antes a esta nueva forma de contar historias; por el otro, el uso de estas herramientas por parte de lectores y escritores proporciona una realimentación necesaria para la mejora de las herramientas disponibles, haciendo que el proceso de desarrollo hacia la madurez de la hiperficción sea más rápido y constante.

6.2 Trabajos futuros

Existen varios aspectos en los que se puede continuar trabajando tanto en HyperAuthor como en HyperViewer. En este documento se van a hacer algunas sugerencias en cuanto a trabajos futuros en dos líneas diferentes: por un lado, mejoras a la aplicación actual y por otro, modificaciones interesantes que podrían realizarse sobre la misma. Se va a abordar la exposición de los posibles trabajos futuros de la misma forma que se ha venido haciendo a lo largo de todo el documento: de manera independiente para HyperAuthor e HyperViewer.

6.2.1 HyperAuthor

El actual HyperAuthor tiene algunas limitaciones de uso. Se podría hacer de esta herramienta un entorno de creación algo más completo y sofisticado implementando las siguientes mejoras en la aplicación.

- En primer lugar, la mayoría de las posibilidades de personalización que ofrece la herramienta están destinadas a "aliviar" el área de dibujo de la estructura hipertextual ya que, cuando se trabaja con hipertextos complejos, ésta se sobrecarga en exceso. Una mejor solución para evitar este inconveniente sería que los bucles abiertos y secuencias satélite pudieran "empaquetarse" gráficamente en un único nodo y que al pinchar sobre él se desplegase. De esta forma la visualización del diagrama sería mucho más limpia y además habría espacio para estructuras mayores.

- Implementar, siguiendo esta misma línea de razonamiento, una función de zoom que permita una mayor flexibilidad de visualización de la estructura hipertextual y por lo tanto también una mayor comodidad a la hora de trabajar en la herramienta.

- Permitir que los nombres de los nodos y los enlaces pudiesen aparecer en varias líneas en el área de trabajo de la herramienta en lugar de en una sola como ocurre actualmente también facilitaría la generación de diagramas complejos. Para suplir esta carencia en la versión definitiva de la herramienta se incluyeron los "*tooltip*", pero parece interesante poder dar un paso más allá.

- De la misma manera y con respecto a los diagramas complejos, su edición se facilitaría si los nombres de nodos y ejes pudiesen editarse en el propio diagrama y no solo cambiar desde dentro del propio editor de nodos.

- Implementar la posibilidad de que se puedan usar imágenes como enlaces. Esto es, que desde el editor de nodos se permita la opción de incluir una imagen en el texto y hacer que esta imagen actúe como enlace hacia otro nodo.

- Mejorar el editor de nodos aumentando el grado de adaptación a funciones específicas de la edición de nodos eliminando las posibles desventajas de la utilización de un lenguaje de base tan específico como el HTML. En esta línea se incluiría por ejemplo la implementación de una función que permita incluir márgenes y tabulaciones en el texto.

- Permitir el uso del *drag&drop* en la herramienta para funciones como las de abrir un hipertexto guardado en disco. Esto implicaría que la herramienta debería ser capaz de abrir un hipertexto si el usuario ha arrastrado el fichero que lo contiene al área de trabajo del mismo modo que si el usuario hubiese utilizado el botón destinado a tal efecto en la barra de herramientas.

- Incluir un mecanismo que facilite la tarea de guardado o cerrado de la herramienta cuando hay muchos nodos en edición abiertos en ella. En este caso, HyperAuthor pregunta uno a uno, por todos los nodos abiertos con cambios sin guardar, si estos cambios deben o no ser guardados. La tarea de contestar a la herramienta sobre cada uno de esos nodos puede resultar demasiado engorrosa, por lo que la implementación de una pequeña pantalla que aparezca en estos casos y que incluya una lista con todos los nodos con cambios sin guardar facilitaría mucho esta tarea. El usuario tendría simplemente que marcar con un tick aquellos nodos que quiera guardar y pulsar en aceptar.

- Finalmente, añadir a la aplicación la posibilidad de dar soporte a capítulos dentro de su propia estructura e incluso de permitir enlaces a nodos pertenecientes a otro hipertexto haría que la herramienta fuese más completa. De esta forma, existiría un nodo de inicio de capítulo y otro de fin, marcados de forma especial y que serían los únicos que tendrían la capacidad de enlazar con

nodos "externos" (entendido esto como nodos ubicados en otros capítulos o en otros hipertextos).

Sin embargo, no se debe caer en la tentación de añadir demasiadas cosas a HyperAuthor, ya que uno de sus requisitos fundamentales es mantener una interfaz y una funcionalidad lo más sencilla y atrayente posible de manera que sea fácilmente utilizada por personas pertenecientes al mundo de las Humanidades a las que no hay por qué presuponer conocimientos técnicos avanzados. En este sentido podría resultar interesante la inclusión de un manual de ayuda extenso y fácilmente accesible en la aplicación.

Por otro lado, ya se ha mencionado con anterioridad en este documento que las obras creadas mediante el uso de HyperAuthor deben ir únicamente un poco más allá de aquellas que forman parte de la literatura tradicional. Por este motivo no es conveniente incluir funcionalidades que tengan como resultado un impacto demasiado grande en las obras creadas hasta que el público se haya habituado a este tipo de literatura. La herramienta deberá permanecer, por lo tanto, en constante evolución para cubrir las necesidades y expectativas de los lectores a medida que estos se hagan usuarios más frecuentes de literatura hipertextual.

Independientemente de estas mejoras, la idea de HyperAuthor podría transformarse para introducirse en ámbitos distintos. Por un lado, sería interesante transformar HyperAuthor en una aplicación WEB 2.0. Las tecnologías actuales tienen recursos suficientes para conseguir las capacidades gráficas necesarias, y la aplicación se vería beneficiada de las ventajas de correr sobre la red (ubicuidad, accesibilidad a un público mucho más numeroso, eliminación de la necesidad de una máquina virtual instalada en el equipo del usuario, etc).

6.2.2 HyperViewer

En lo referente a HyperViewer, actualmente es una herramienta de visualización bastante básica que se ha centrado en la aplicación de algunos principios de la hipertextualidad y las características necesarias para lograr una lectura profunda y de calidad con este tipo de literatura. HyperViewer tiene, por lo tanto, algunas limitaciones de uso en cuanto a funcionalidades que se presuponen en una herramienta de lectura. Se podría hacer de esta herramienta un lector mucho más sofisticado con la inclusión de las siguientes funcionalidades adicionales.

- Permitir el uso de un "separador" para que el usuario lector pueda marcar la página en la que detuvo la lectura.
- Permitir la inclusión de información externa por parte del usuario a modo de notas.
- Incluir marcadores o *bookmarks* que permitan al usuario volver a un cierto punto seleccionado en la obra.

- Implementar motores de búsqueda con diferentes técnicas de recuperación de información. En este sentido, sería útil también la inclusión de campos o etiquetas que faciliten la recuperación de la información.

- Añadir una nueva funcionalidad que permita crear una representación gráfica de la red completa de hipertexto en la que se indique el lugar donde se encuentra el usuario. Además estaría permitida la navegación usando esa representación gráfica en la que se muestren los nodos existentes y/o recorridos así como las relaciones con otros nodos y si se han visitado o no.

- Incluir suaves efectos de transición que acompañen al hecho de pasar página tras un clic de usuario. Estos "miniefectos" deberían estar limitados a fundidos y disoluciones, siendo siempre el mismo para no distraer la atención del lector.

Con independencia de todas estas mejoras, sería realmente útil poder utilizar este visor no sólo como una herramienta local. La transformación de HyperViewer a un *applet* que reprodujera el contenido de un determinado hipertexto permitiría su inclusión en páginas web, haciendo de esta forma su acceso más fácil y global. En este caso se trataría de un visor on-line que permitiría a un autor colgar su obra en Internet y permitir además su visualización a través del mismo medio.

En este caso, sería factible y además muy interesante dar un paso más allá y dar paso a la creación de repositorios web donde se pudiera almacenar este nuevo tipo de literatura. De esta manera se crearían bibliotecas de hipertexto que además podrían reproducirse en línea.

Una vez alcanzado este grado de madurez de la hiperficción en el que los repositorios fueran una realidad, seria necesaria en el visor la creación de diferentes perfiles de uso atendiendo a los privilegios que cada usuario pueda tener sobre la obra.

Bibliografía

[1] ALAYÓN GÓMEZ, Jerónimo. *"Perspectiva y problemas de la narrativa hipertextual"*. 2006 http://www.cibersociedad.net/congres2006/gts/comunicacio.php?id=833&llengua=e

[2] AUERBACH, Sarah. *"Ficción hipertextual: Una teoría literaria"*.1995 Traducción de Susana Pajares Toska 1997. http://www.ucm.es/info/especulo/hipertul/auerbach.html

[3] BECKER, Howard S. *"A New Art Form: Hypertext Fiction"* http://home.earthlink.net/~hsbecker/lisbon.html http://www.uv.es/~fores/programa/becker_hypertextfiction.html

[4] BIRKERTS, Sven. *"Elegía a Gutemberg: el futuro de la lectura en la era electrónica"* 1999

[5] EASTGATE. http://www.eastgate.com

[6] ESPÉCULO. *Revista de Estudios Literarios* http://www.ucm.es/OTROS/especulo/

[7] ESTRADA, Socorro. *"Internet, un espacio Nuevo y sin límites para la creación literaria"* 2005. http://bloggy.com.ar/drbloggy/archive/2005/01/02/300.aspx

[8] FAUTH, Jurgen. *"Poles in your FACE: The Promises and Pitfalls of Hyperfiction"*. 1995 http://www.uoc.edu/in3/hermeneia/sala_de_lectura/j_fauth_poles_in_your_face.htm

[9] GARCÍA RUEDA, José Jesús y FRANCO ESPINOSA, Carolina. *"Narrativa hipermediática:"* http://www.it.uc3m.es/rueda/Madrid05/MundoInternet05.htm

[10] GARCÍA RUEDA, José Jesús y FRANCO ESPINOSA, Carolina. *"Narrativa hipermediática: los nuevos contenidos para el cibermundo"*. http://www.cibersociedad.net/congres2004/grups/fitxacom_publica2.php?grup=60&id=508&idioma=es

[11] GÓMEZ TRUEBA, Teresa. *"De la narrativa posmoderna a la hiperficción"* http://www.wikilearning.com/de_la_narrativa_posmoderna_a_la_hiperficcion-wkccp-17921-1.htm

[12] HIPERTULIA. http://www.ucm.es/info/especulo/hipertul/index.htm

[13] JOYCE, James. "*Finnegans Wake"* 1939

[14] JOYCE, Michael. *"Introduction: The Comfort of Knowing We Are Not Lost"* en *Of Two Minds: Hypertext Pedagogy and Poetics* 1955

http://www.press.umich.edu/pdf/0472095781-intro.html

[15] LAMARCA LAPUENTE, Mª Jesús. *"Hipertexto. El nuevo concepto de documento en la cultura de la imagen"*
http://www.hipertexto.info/

[16] LANDOW, George P. *"Hipertexto. La convergencia de la teoría crítica contemporánea y la tecnología"*. Barcelona, Paidós, 1992 (1995)

[17] MURRAY, Janet H. *"Hamlet en la holocubierta"*. Traducción de Susana Pajares Toska. Barcelona, Paidós, 1999

[18]NEUBURG,Matt. *"Light Your Fire with Tinderbox"*. Publicado en TidBits.
http://db.tidbits.com/article/6959

[19] NEUBURG, Matt. *"Tell me a Storyspace"*. Publicado en TidBits.
http://db.tidbits.com/issue/582

[20] O'BRIEN, Flann. *"At Swim two Birds"*. 1939

[21] ORIHUELA, José Luis. *"El narrador en ficción interactiva"*.
http://www.ucm.es/info/especulo/hipertul/califia.htm

[22] PAJARES TOSKA, Susana. "*Las posibilidades de la narrativa hipertextual*" 1997.
http://www.ucm.es/info/especulo/numero6/s_pajare.htm

[23] PASTOR, Juan Antonio. "*La escritura hipermedia*"
http://www.ucm.es/info/multidoc/multidoc/revista/cuad6-7/saorin.htm

[24] PUENTE LAMARCA, María Jesús. *"Hipertexto: El Nuevo concepto de documento en la cultura de la imagen"*
http://www.hipertexto.info

[25] RODRÍGUEZ DE LAS HERAS, Antonio. *"El libro digital"*. Barcelona, 1999
http://www.uoc.edu/web/esp/articles/digitum_art_eras.html

[26] RODRÍGUEZ DE LAS HERAS, Antonio. *"Navegar por la información"*. Fundesco, 1991

[27] DEEMER, Charles. http://www.ibiblio.org/cdeemer/Essays.htm

[28] Patrón MVC. Estándar http://java.sun.com/blueprints/patterns/MVC.html

[29] Patrón MVC. Página relacionada.
http://www.proactiva-calidad.com/java/patrones/mvc.html

[30] JOYCE, Michael. *"Afternoon, a story"* Eastgate, 1987
http://www.eastgate.com/catalog/Afternoon.html

[31] Documentación de clases de Swing
http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html

[32] KRUTSCH, Kenneth F., CARGO David S. y HOWLETT Virginia. *"Professional Java Custom UI Components"* Ed. Wrox.

[33] Documentación de clases de JGraph
http://www.jgraph.com/doc/jgraph

[34] Homepage de SimplyHTML
http://www.lightdev.com/page/3.htm
http://sourceforge.net/projects/simplyhtml/

[35] Documentación de arquitectura de Swing. *"A Swing Architecture OverView"*. Sun Microsystems.
http://java.sun.com/products/jfc/tsc/articles/architecture/

[36] Preguntas frecuentes sobre Jgraph.
http://www.jgraph.com/faq.html?show=3.11

[37] Documentación sobre JDom
http://www.jdom.org/docs/apidocs/

[38] API de Java
http://java.sun.com/j2se/1.5.0/docs/api

[39] Manual sobre Java, XML y JDom
http://www.javaworld.com/javaworld/jw-05-2000/jw-0518-jdom.html

[40] Manual sobre JTree
http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html

[41] FOUCAULT, Michel. *"The Archeology of Knowledge"*. New York. Harper and Row, 1976.

[42] DEEMER, Charles. *"Chateau de Mort"*, 1996.

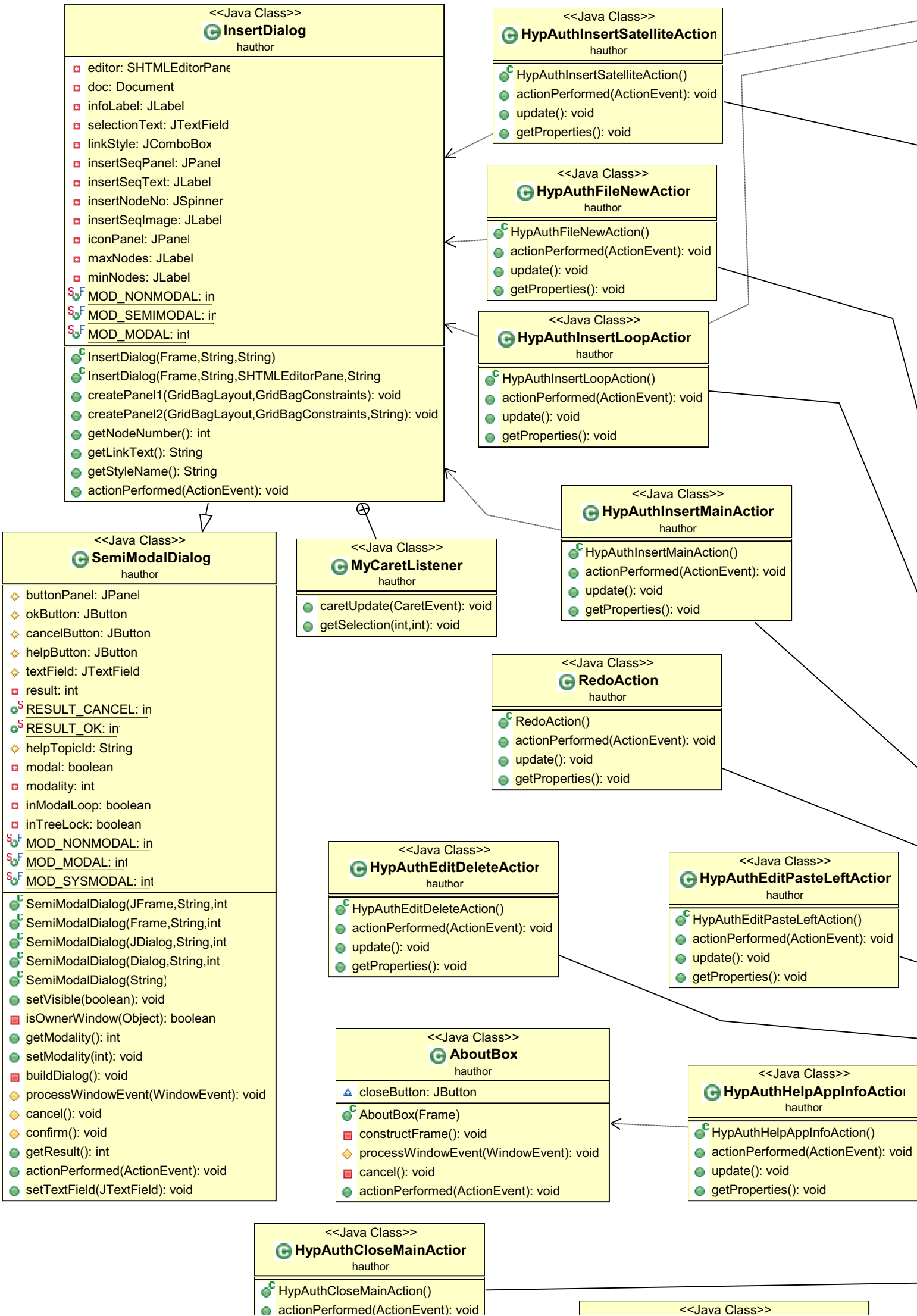
[43]WEISS, Sebastian, MÜLLER, Wolfgang, SPIERLING, Ulrike y STEIMLE, Florian. *"Scenejo, An Interactive Storytelling Platform"*. Springer-Verlag Berlin Heidelberg 2005

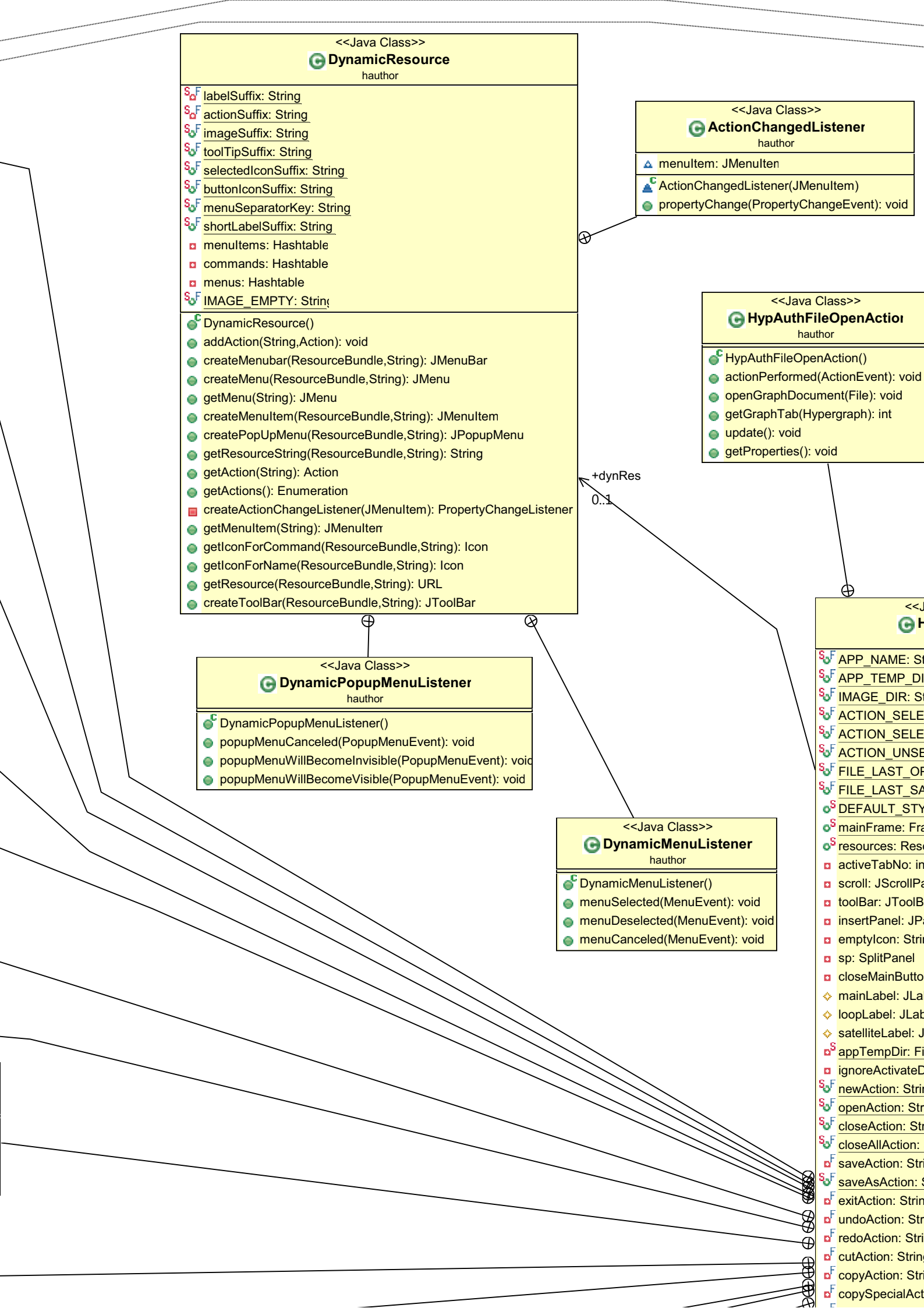
[44] *"Cuéntame un cuento (StorySpace)"*
http://www.tidbits.com/tb-issues/lang/es/TidBITS-es-582.html#lnk3

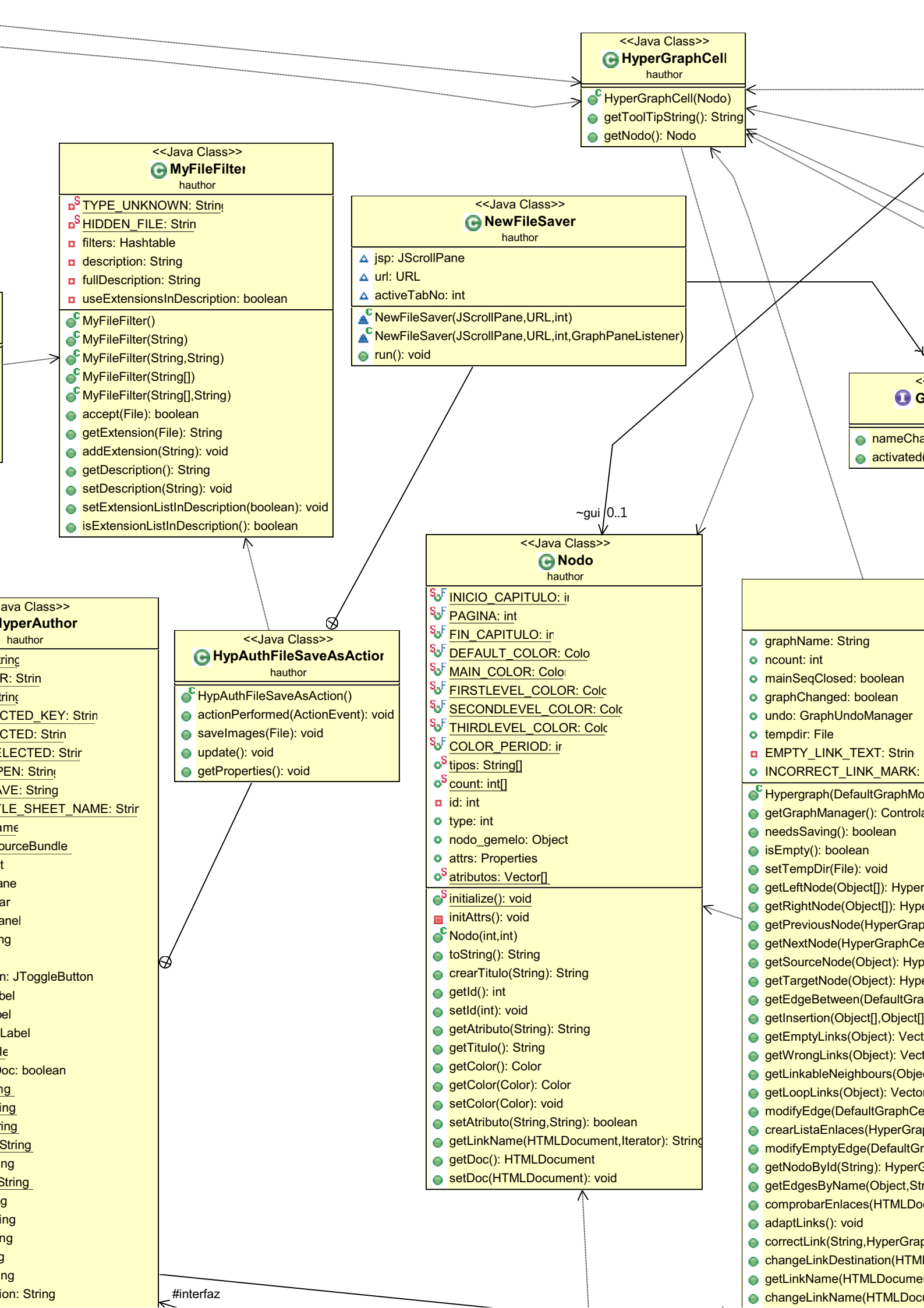
[45] SÁEZ VACAS, F. *"Nuevos paradigmas empresariales y tecnológicos"* Telos, cuadernos de comunicación, tecnología y sociedad. Nº44. Madrid. 1996

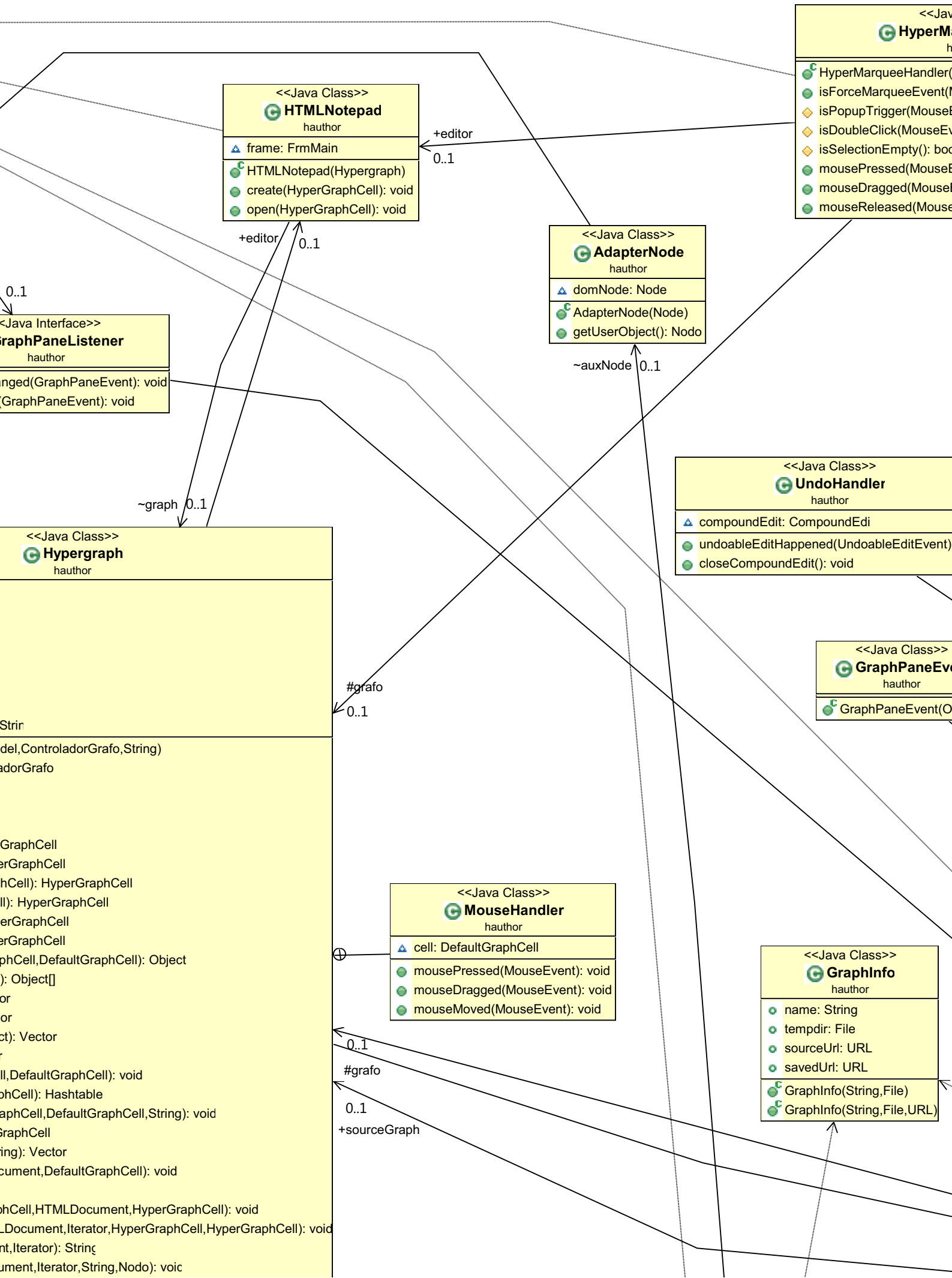
Pliego II
Planos

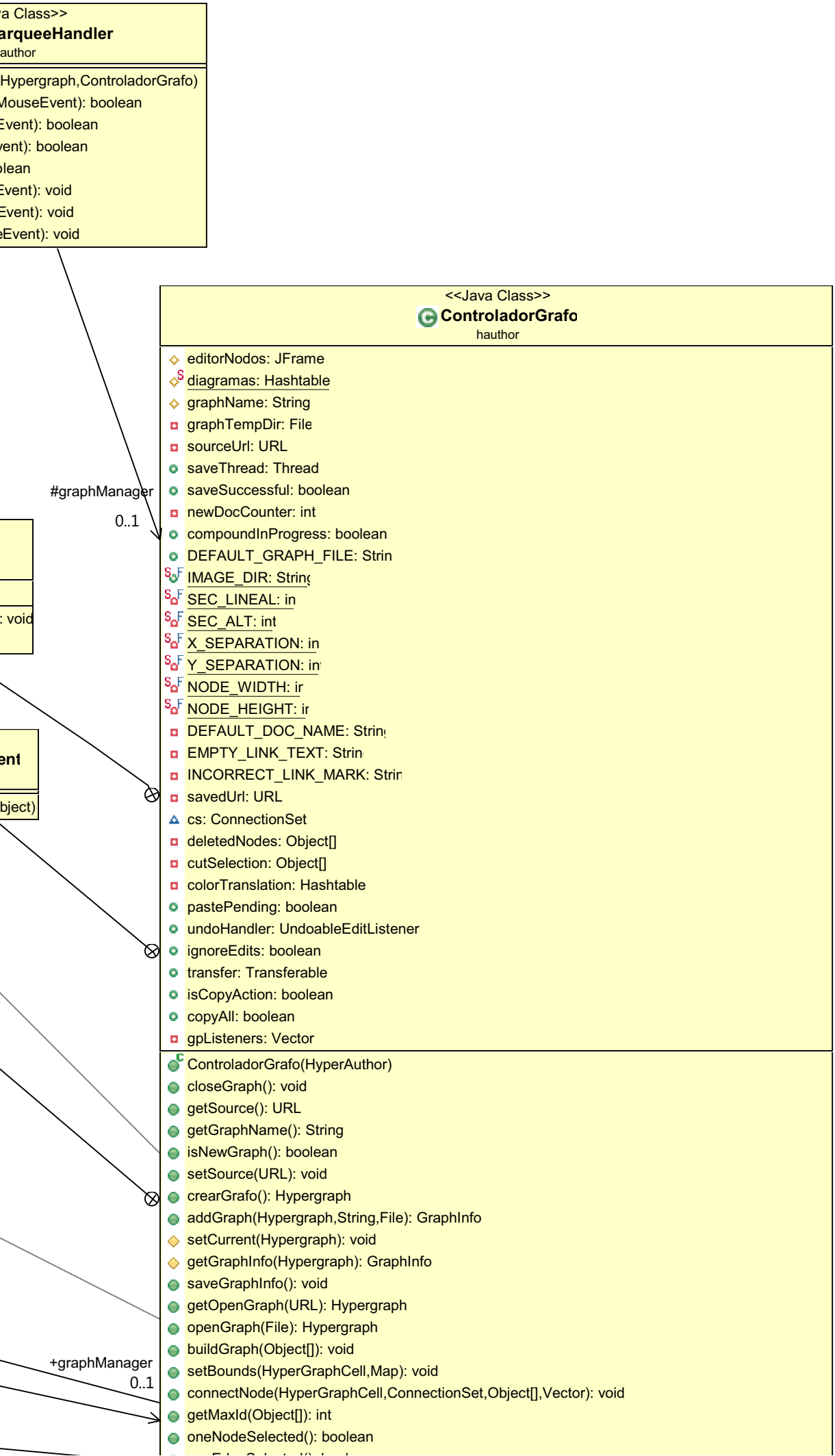
HyperAuthor
Planos UML

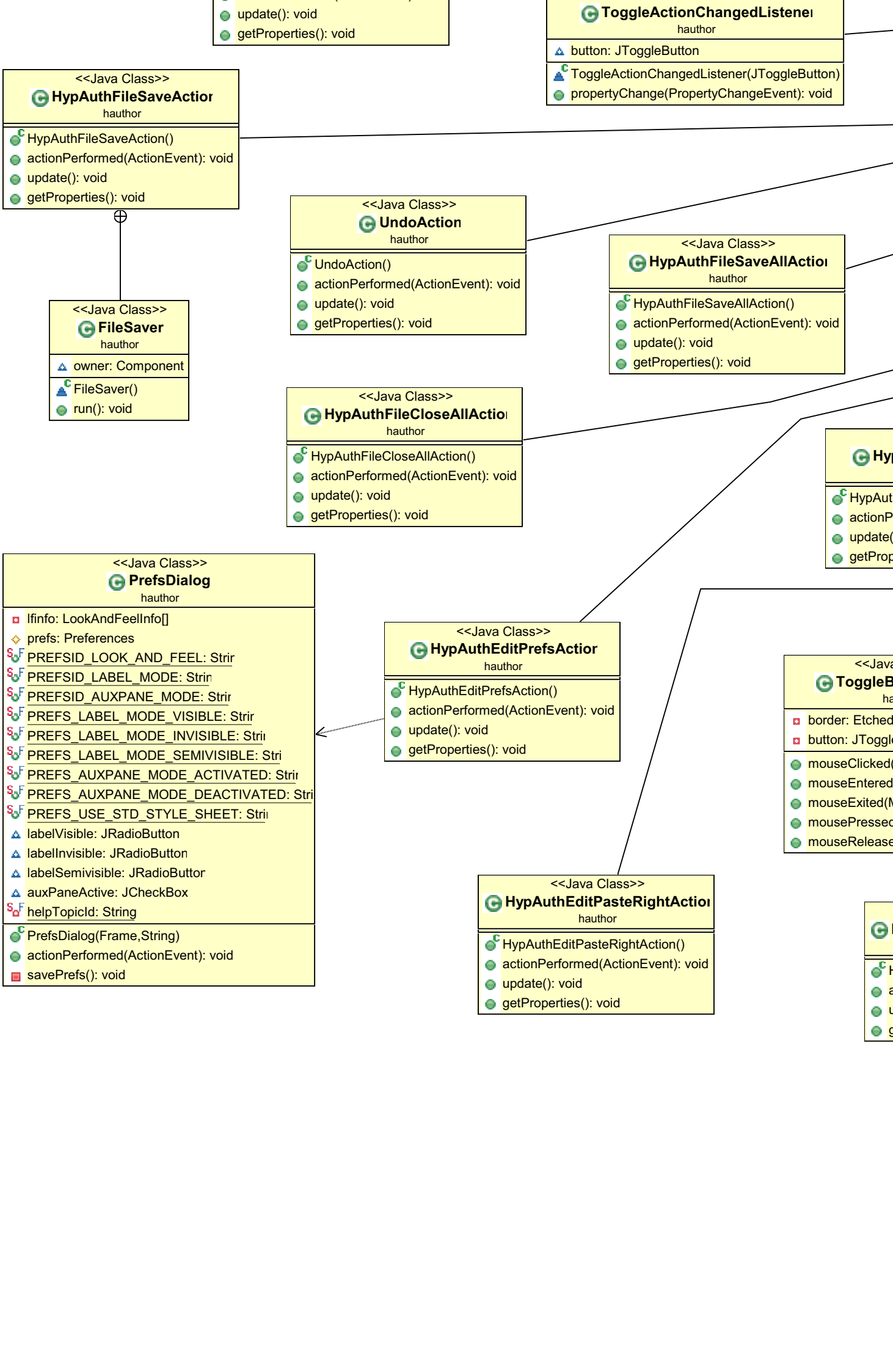

<<Java Class>>
InsertDialog
hauthor
editor: SHTMLEditorPane
doc: Document
infoLabel: JLabel
selectionText: JTextField
linkStyle: JComboBox
insertSeqPanel: JPanel
insertSeqText: JLabel
insertNodeNo: JSpinner
insertSeqImage: JLabel
iconPanel: JPanel
maxNodes: JLabel
minNodes: JLabel
MOD_NONMODAL: in
MOD_SEMIMODAL: in
MOD_MODAL: int
InsertDialog(Frame,String,String)
InsertDialog(Frame,String,SHTMLEditorPane,String
createPanel1(GridBagLayout,GridBagConstraints): void
createPanel2(GridBagLayout,GridBagConstraints,String): void
getNodeNumber(): int
getLinkText(): String
getStyleName(): String
actionPerformed(ActionEvent): void
<<Java Class>>
HypAuthInsertSatelliteAction
hauthor
HypAuthInsertSatelliteAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthFileNewActior
hauthor
HypAuthFileNewAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthInsertLoopActior
hauthor
HypAuthInsertLoopAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthInsertMainActior
hauthor
HypAuthInsertMainAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
SemiModalDialog
hauthor
buttonPanel: JPanel
okButton: JButton
cancelButton: JButton
helpButton: JButton
textField: JTextField
result: int
RESULT_CANCEL: in
RESULT_OK: in
helpTopicId: String
modal: boolean
modality: int
inModalLoop: boolean
inTreeLock: boolean
MOD_NONMODAL: in
MOD_MODAL: int
MOD_SYSMODAL: int
SemiModalDialog(JFrame,String,int
SemiModalDialog(Frame,String,int
SemiModalDialog(JDialog,String,int
SemiModalDialog(Dialog,String,int
SemiModalDialog(String)
setVisible(boolean): void
isOwnerWindow(Object): boolean
getModality(): int
setModality(int): void
buildDialog(): void
processWindowEvent(WindowEvent): void
cancel(): void
confirm(): void
getResult(): int
actionPerformed(ActionEvent): void
setTextField(JTextField): void
<<Java Class>>
MyCaretListener
hauthor
caretUpdate(CaretEvent): void
getSelection(int,int): void
<<Java Class>>
RedoAction
hauthor
RedoAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthEditDeleteActior
hauthor
HypAuthEditDeleteAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthEditPasteLeftActior
hauthor
HypAuthEditPasteLeftAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
AboutBox
hauthor
closeButton: JButton
AboutBox(Frame)
constructFrame(): void
processWindowEvent(WindowEvent): void
cancel(): void
actionPerformed(ActionEvent): void
<<Java Class>>
HypAuthHelpAppInfoActio
hauthor
HypAuthHelpAppInfoAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthCloseMainActior
hauthor
HypAuthCloseMainAction()
actionPerformed(ActionEvent): void
<<Java Class>

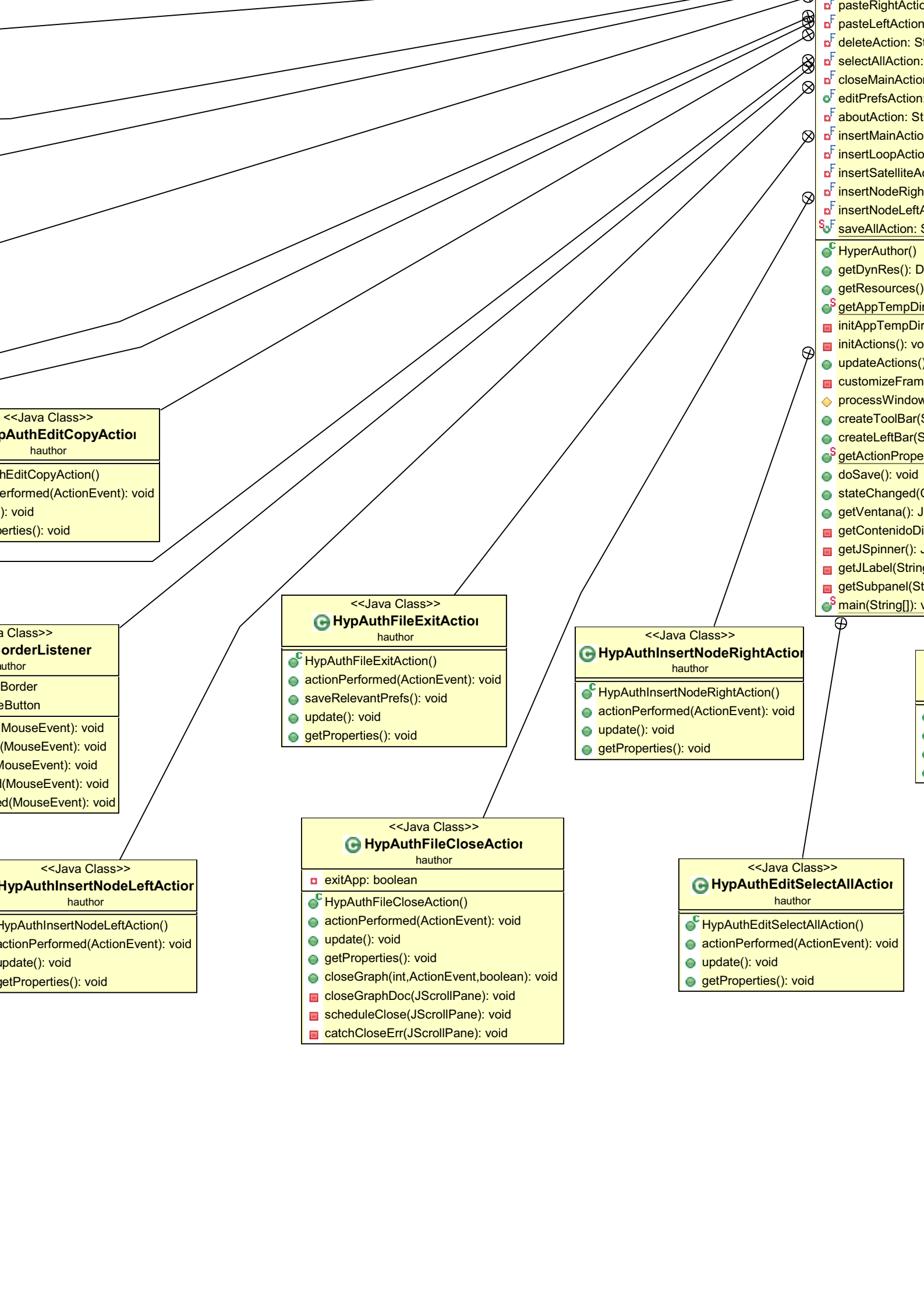

<<Java Class>>
DynamicResource
hauthor
labelSuffix: String
actionSuffix: String
imageSuffix: String
toolTipSuffix: String
selectedIconSuffix: String
buttonIconSuffix: String
menuSeparatorKey: String
shortLabelSuffix: String
menuItems: Hashtable
commands: Hashtable
menus: Hashtable
IMAGE_EMPTY: Strin
DynamicResource()
addAction(String,Action): void
createMenubar(ResourceBundle,String): JMenuBar
createMenu(ResourceBundle,String): JMenu
getMenu(String): JMenu
createMenuItem(ResourceBundle,String): JMenuItem
createPopUpMenu(ResourceBundle,String): JPopupMenu
getResourceString(ResourceBundle,String): String
getAction(String): Action
getActions(): Enumeration
createActionChangeListener(JMenuItem): PropertyChangeListener
getMenuItem(String): JMenuItem
getIconForCommand(ResourceBundle,String): Icon
getIconForName(ResourceBundle,String): Icon
getResource(ResourceBundle,String): URL
createToolBar(ResourceBundle,String): JToolBar
<<Java Class>>
ActionChangedListener
hauthor
menuItem: JMenuItem
ActionChangedListener(JMenuItem)
propertyChange(PropertyChangeEvent): void
<<Java Class>>
HypAuthFileOpenAction
hauthor
HypAuthFileOpenAction()
actionPerformed(ActionEvent): void
openGraphDocument(File): void
getGraphTab(Hypergraph): int
update(): void
getProperties(): void
+dynRes
0..1
<<Java Class>>
DynamicPopupMenuListener
hauthor
DynamicPopupMenuListener()
popupMenuCanceled(PopupMenuEvent): void
popupMenuWillBecomeInvisible(PopupMenuEvent): void
popupMenuWillBecomeVisible(PopupMenuEvent): void
<<Java Class>>
DynamicMenuListener
hauthor
DynamicMenuListener()
menuSelected(MenuEvent): void
menuDeselected(MenuEvent): void
menuCanceled(MenuEvent): void

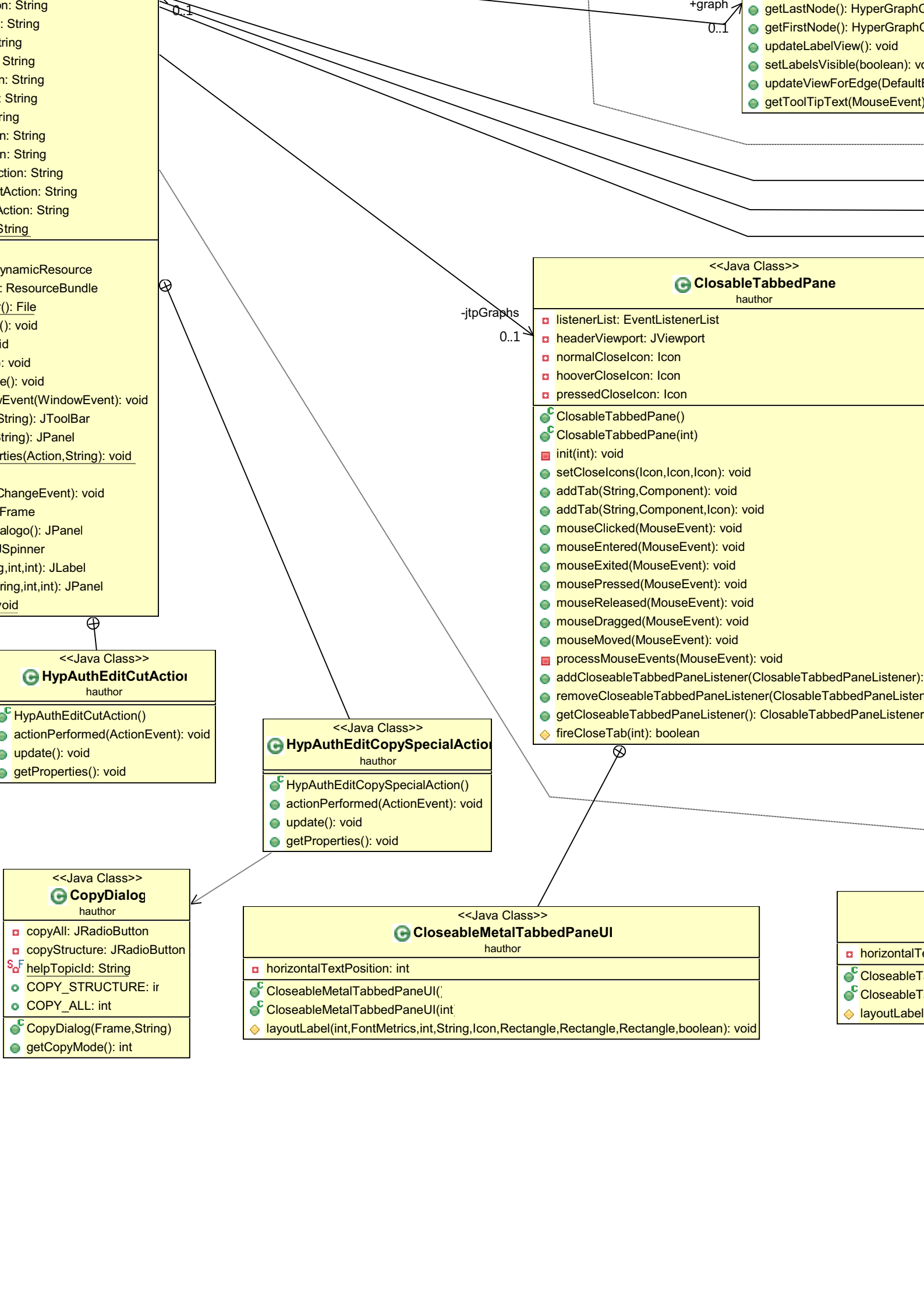

<<Java Class>>
HyperGraphCell
hauthor
HyperGraphCell(Nodo)
getToolTipString(): String
getNodo(): Nodo
<<Java Class>>
MyFileFilter
hauthor
TYPE_UNKNOWN: String
HIDDEN_FILE: Strin
filters: Hashtable
description: String
fullDescription: String
useExtensionsInDescription: boolean
MyFileFilter()
MyFileFilter(String)
MyFileFilter(String,String)
MyFileFilter(String[])
MyFileFilter(String[],String)
accept(File): boolean
getExtension(File): String
addExtension(String): void
getDescription(): String
setDescription(String): void
setExtensionListInDescription(boolean): void
isExtensionListInDescription(): boolean
<<Java Class>>
NewFileSaver
hauthor
jsp: JScrollPane
url: URL
activeTabNo: int
NewFileSaver(JScrollPane,URL,int)
NewFileSaver(JScrollPane,URL,int,GraphPaneListener)
run(): void
~gui 0..1
<<Java Class>>
Nodo
hauthor
INICIO_CAPITULO: i
PAGINA: int
FIN_CAPITULO: in
DEFAULT_COLOR: Colo
MAIN_COLOR: Colo
FIRSTLEVEL_COLOR: Colo
SECONDLEVEL_COLOR: Colo
THIRDLEVEL_COLOR: Colo
COLOR_PERIOD: in
tipos: String[]
count: int[]
id: int
type: int
nodo_gemelo: Object
attrs: Properties
atributos: Vector[]
initialize(): void
initAttrs(): void
Nodo(int,int)
toString(): String
crearTitulo(String): String
getId(): int
setId(int): void
getAtributo(String): String
getTitulo(): String
getColor(): Color
getColor(Color): Color
setColor(Color): void
setAtributo(String,String): boolean
getLinkName(HTMLDocument,Iterator): String
getDoc(): HTMLDocument
setDoc(HTMLDocument): void
<<Java Class>>
HypAuthFileSaveAsAction
hauthor
HypAuthFileSaveAsAction()
actionPerformed(ActionEvent): void
saveImages(File): void
update(): void
getProperties(): void
graphName: String
ncount: int
mainSeqClosed: boolean
graphChanged: boolean
undo: GraphUndoManager
tempdir: File
EMPTY_LINK_TEXT: Strin
INCORRECT_LINK_MARK:
Hypergraph(DefaultGraphMo
getGraphManager(): Controla
needsSaving(): boolean
isEmpty(): boolean
setTempDir(File): void
getLeftNode(Object[]): Hyper
getRightNode(Object[]): Hype
getPreviousNode(HyperGrap
getNextNode(HyperGraphCe
getSourceNode(Object): Hyp
getTargetNode(Object): Hype
getEdgeBetween(DefaultGra
getInsertion(Object[],Object[]
getEmptyLinks(Object): Vect
getWrongLinks(Object): Vect
getLinkableNeighbours(Objec
getLoopLinks(Object): Vecto
modifyEdge(DefaultGraphCe
crearListaEnlaces(HyperGra
modifyEmptyEdge(DefaultGr
getNodoById(String): HyperG
getEdgesByName(Object,Str
comprobarEnlaces(HTMLDo
adaptLinks(): void
correctLink(String,HyperGrap
changeLinkDestination(HTML
getLinkName(HTMLDocume
changeLinkName(HTMLDocu
#interfaz

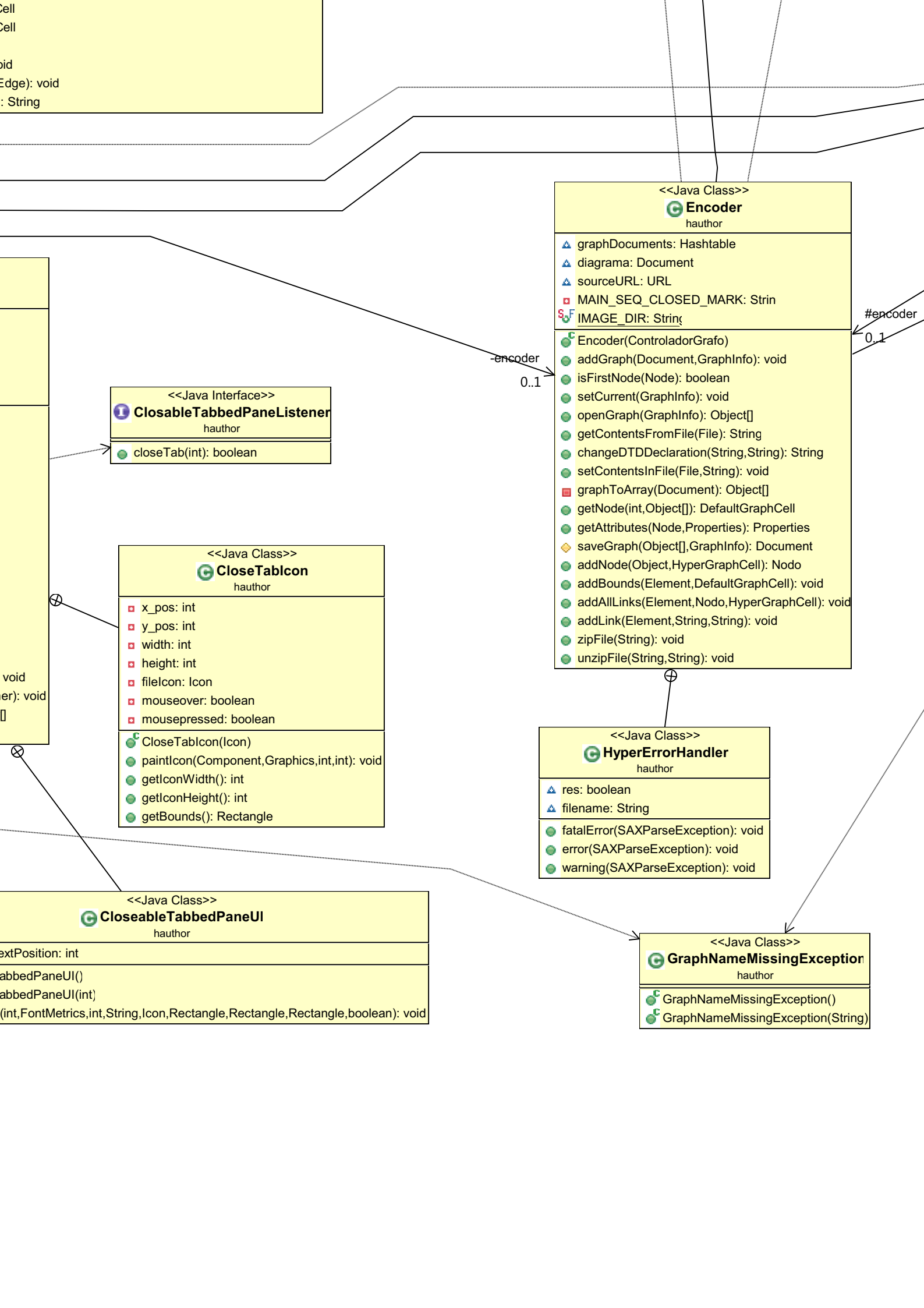

<<Java Class>>
HTMLNotepad
hauthor
frame: FrmMain
HTMLNotepad(Hypergraph)
create(HyperGraphCell): void
open(HyperGraphCell): void
+editor
0..1
+editor
0..1
<<Java Class>>
AdapterNode
hauthor
domNode: Node
AdapterNode(Node)
getUserObject(): Nodo
~auxNode
0..1
0..1
<<Java Interface>>
GraphPaneListener
hauthor
nged(GraphPaneEvent): void
(GraphPaneEvent): void
~graph
0..1
<<Java Class>>
Hypergraph
hauthor
Strir
del,ControladorGrafo,String)
adorGrafo
GraphCell
erGraphCell
hCell): HyperGraphCell
ll): HyperGraphCell
erGraphCell
erGraphCell
phCell,DefaultGraphCell): Object
): Object[]
or
or
ct): Vector
ll,DefaultGraphCell): void
hCell): Hashtable
aphCell,DefaultGraphCell,String): void
GraphCell
ring): Vector
cument,DefaultGraphCell): void
hCell,HTMLDocument,HyperGraphCell): void
LDocument,Iterator,HyperGraphCell,HyperGraphCell): void
nt,Iterator): String
ument,Iterator,String,Nodo): voic
#grafo
0..1
<<Java Class>>
UndoHandler
hauthor
compoundEdit: CompoundEdi
undoableEditHappened(UndoableEditEvent)
closeCompoundEdit(): void
<<Java Class>>
GraphPaneEv
hauthor
GraphPaneEvent(O
<<Java Class>>
MouseHandler
hauthor
cell: DefaultGraphCell
mousePressed(MouseEvent): void
mouseDragged(MouseEvent): void
mouseMoved(MouseEvent): void
0..1
#grafo
0..1
+sourceGraph
<<Java Class>>
GraphInfo
hauthor
name: String
tempdir: File
sourceUrl: URL
savedUrl: URL
GraphInfo(String,File)
GraphInfo(String,File,URL)
<<Ja
HyperM
h
HyperMarqueeHandler(
isForceMarqueeEvent(
isPopupTrigger(Mouse
isDoubleClick(MouseEv
isSelectionEmpty(): boo
mousePressed(Mouse
mouseDragged(Mouse
mouseReleased(Mouse


a Class>>
arqueeHandler
author
Hypergraph,ControladorGrafo)
MouseEvent): boolean
Event): boolean
vent): boolean
olean
Event): void
Event): void
eEvent): void
<<Java Class>>
ControladorGrafo
hauthor
editorNodos: JFrame
diagramas: Hashtable
graphName: String
graphTempDir: File
sourceUrl: URL
saveThread: Thread
saveSuccessful: boolean
newDocCounter: int
compoundInProgress: boolean
DEFAULT_GRAPH_FILE: Strin
IMAGE_DIR: String
SEC_LINEAL: in
SEC_ALT: int
X_SEPARATION: in
Y_SEPARATION: in
NODE_WIDTH: ir
NODE_HEIGHT: ir
DEFAULT_DOC_NAME: Strin
EMPTY_LINK_TEXT: Strin
INCORRECT_LINK_MARK: Strir
savedUrl: URL
cs: ConnectionSet
deletedNodes: Object[]
cutSelection: Object[]
colorTranslation: Hashtable
pastePending: boolean
undoHandler: UndoableEditListener
ignoreEdits: boolean
transfer: Transferable
isCopyAction: boolean
copyAll: boolean
gpListeners: Vector
ControladorGrafo(HyperAuthor)
closeGraph(): void
getSource(): URL
getGraphName(): String
isNewGraph(): boolean
setSource(URL): void
crearGrafo(): Hypergraph
addGraph(Hypergraph,String,File): GraphInfo
setCurrent(Hypergraph): void
getGraphInfo(Hypergraph): GraphInfo
saveGraphInfo(): void
getOpenGraph(URL): Hypergraph
openGraph(File): Hypergraph
buildGraph(Object[]): void
setBounds(HyperGraphCell,Map): void
connectNode(HyperGraphCell,ConnectionSet,Object[],Vector): void
getMaxId(Object[]): int
oneNodeSelected(): boolean
#graphManager
0..1
+graphManager
0..1
: void
ent
bject)


update(): void
getProperties(): void
<<Java Class>>
ToggleActionChangedListene
hauthor
button: JToggleButton
ToggleActionChangedListener(JToggleButton)
propertyChange(PropertyChangeEvent): void
<<Java Class>>
HypAuthFileSaveActior
hauthor
HypAuthFileSaveAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
UndoAction
hauthor
UndoAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthFileSaveAllActio
hauthor
HypAuthFileSaveAllAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
FileSaver
hauthor
owner: Component
FileSaver()
run(): void
<<Java Class>>
HypAuthFileCloseAllActio
hauthor
HypAuthFileCloseAllAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
Hyp
HypAut
actionP
update(
getProp
<<Java Class>>
PrefsDialog
hauthor
lfinfo: LookAndFeelInfo[]
prefs: Preferences
PREFSID_LOOK_AND_FEEL: Strir
PREFSID_LABEL_MODE: Strin
PREFSID_AUXPANE_MODE: Strir
PREFS_LABEL_MODE_VISIBLE: Strir
PREFS_LABEL_MODE_INVISIBLE: Stri
PREFS_LABEL_MODE_SEMIVISIBLE: Stri
PREFS_AUXPANE_MODE_ACTIVATED: Strir
PREFS_AUXPANE_MODE_DEACTIVATED: Stri
PREFS_USE_STD_STYLE_SHEET: Stri
labelVisible: JRadioButton
labelInvisible: JRadioButton
labelSemivisible: JRadioButtor
auxPaneActive: JCheckBox
helpTopicId: String
PrefsDialog(Frame,String)
actionPerformed(ActionEvent): void
savePrefs(): void
<<Java Class>>
HypAuthEditPrefsActior
hauthor
HypAuthEditPrefsAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java
ToggleB
ha
border: Etched
button: JToggl
mouseClicked(
mouseEntered
mouseExited(M
mousePressed
mouseRelease
<<Java Class>>
HypAuthEditPasteRightActio
hauthor
HypAuthEditPasteRightAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void

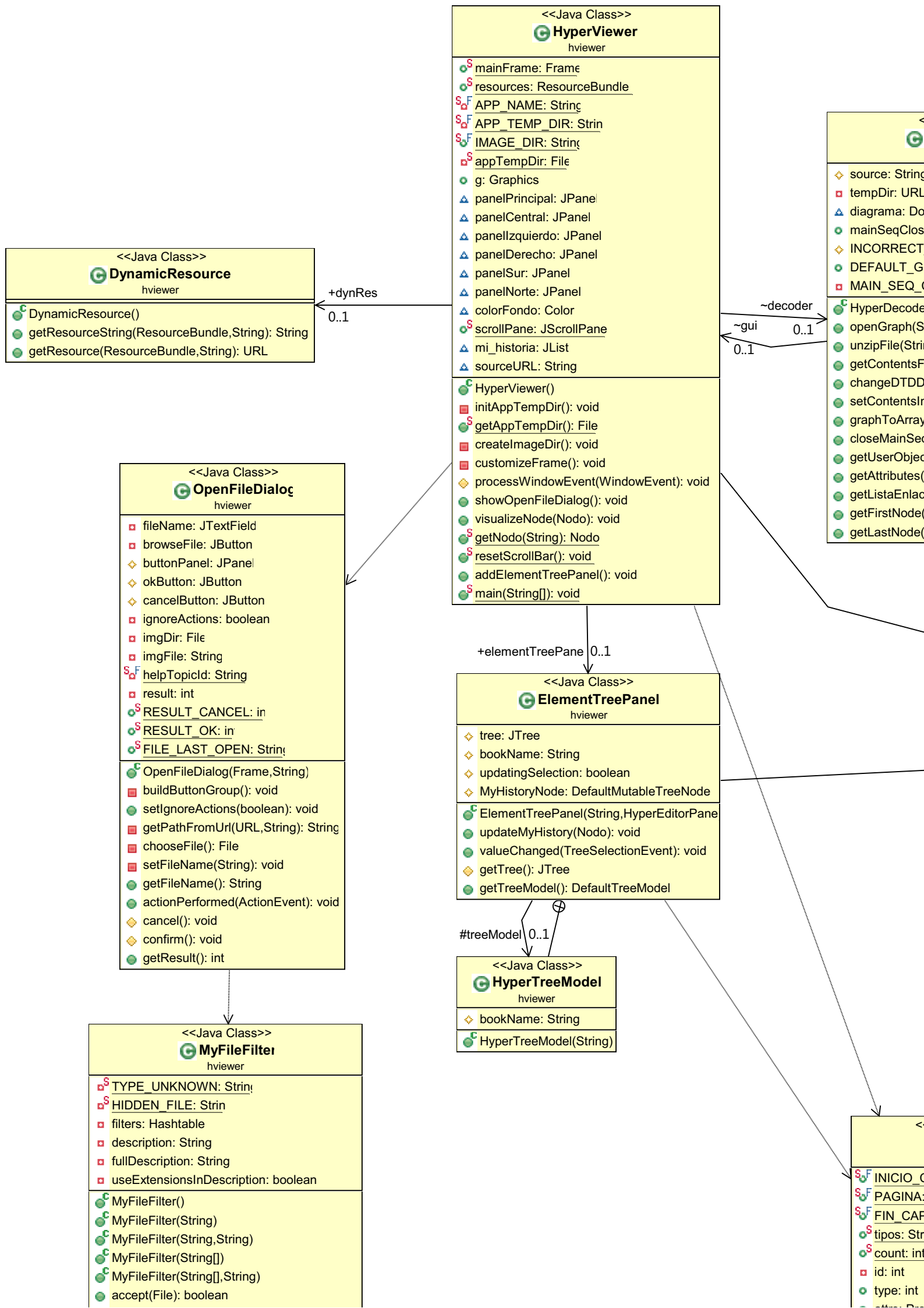

pasteRightActio
pasteLeftAction
deleteAction: St
selectAllAction:
closeMainActio
editPrefsAction
aboutAction: St
insertMainActio
insertLoopActio
insertSatelliteA
insertNodeRigh
insertNodeLeftA
saveAllAction: S
HyperAuthor()
getDynRes(): D
getResources()
getAppTempDir
initAppTempDir
initActions(): vo
updateActions()
customizeFram
processWindow
createToolBar(S
createLeftBar(S
getActionPrope
doSave(): void
stateChanged(
getVentana(): J
getContenidoDi
getJSpinner():
getJLabel(Strin
getSubpanel(St
main(String[]): v
<<Java Class>>
pAuthEditCopyActio
hauthor
hEditCopyAction()
erformed(ActionEvent): void
): void
perties(): void
a Class>>
orderListener
uthor
Border
eButton
MouseEvent): void
(MouseEvent): void
MouseEvent): void
(MouseEvent): void
ed(MouseEvent): void
<<Java Class>>
HypAuthFileExitActio
hauthor
HypAuthFileExitAction()
actionPerformed(ActionEvent): void
saveRelevantPrefs(): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthInsertNodeRightActio
hauthor
HypAuthInsertNodeRightAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthInsertNodeLeftActior
hauthor
HypAuthInsertNodeLeftAction()
ctionPerformed(ActionEvent): void
pdate(): void
getProperties(): void
<<Java Class>>
HypAuthFileCloseActio
hauthor
exitApp: boolean
HypAuthFileCloseAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
closeGraph(int,ActionEvent,boolean): void
closeGraphDoc(JScrollPane): void
scheduleClose(JScrollPane): void
catchCloseErr(JScrollPane): void
<<Java Class>>
HypAuthEditSelectAllActio
hauthor
HypAuthEditSelectAllAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void

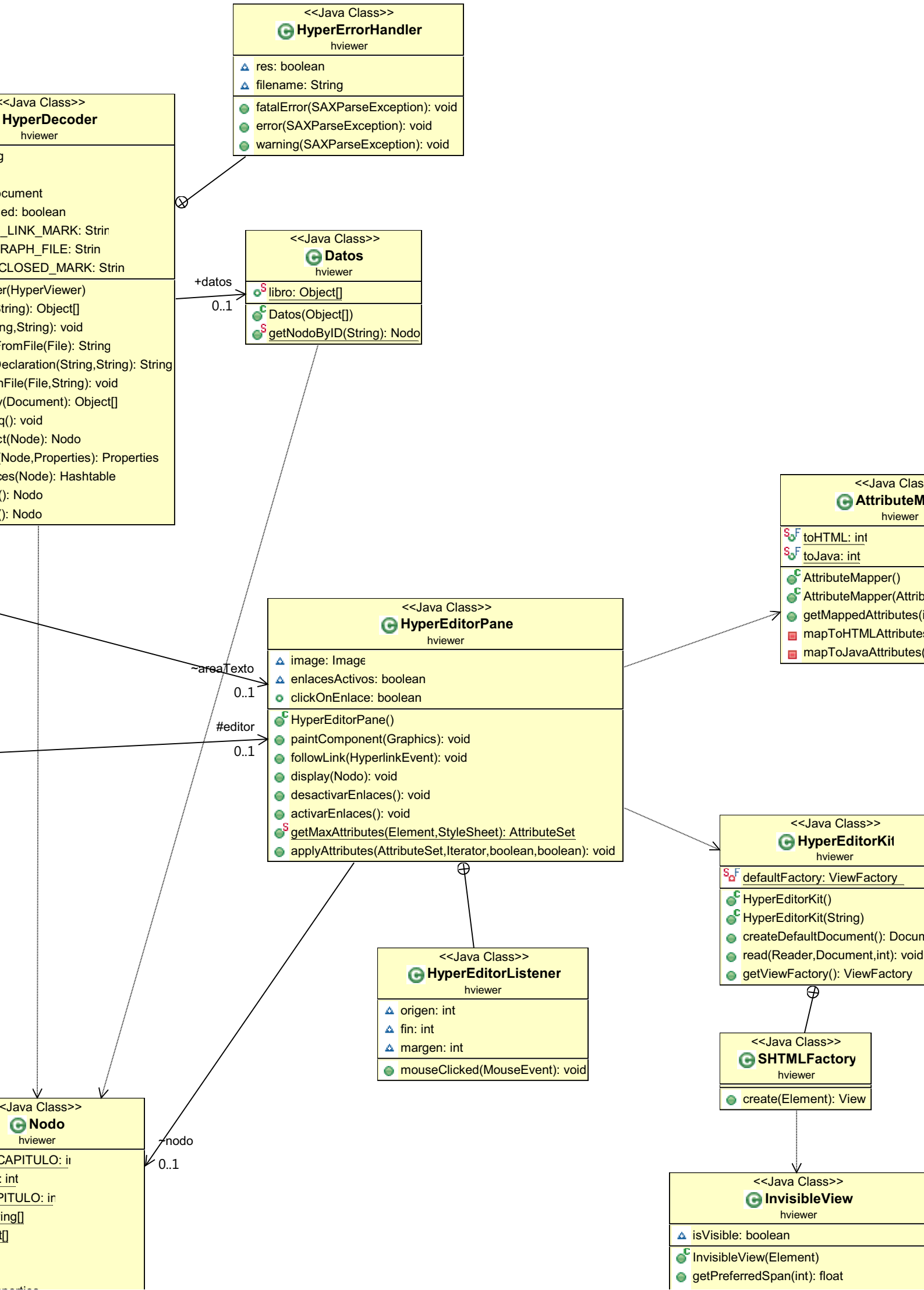

0..1
+graph
0..1
getLastNode(): HyperGraphC
getFirstNode(): HyperGraphC
updateLabelView(): void
setLabelsVisible(boolean): vo
updateViewForEdge(DefaultE
getToolTipText(MouseEvent)
-jtpGraphs
0..1
<<Java Class>>
ClosableTabbedPane
hauthor
listenerList: EventListenerList
headerViewport: JViewport
normalCloseIcon: Icon
hooverCloseIcon: Icon
pressedCloseIcon: Icon
ClosableTabbedPane()
ClosableTabbedPane(int)
init(int): void
setCloseIcons(Icon,Icon,Icon): void
addTab(String,Component): void
addTab(String,Component,Icon): void
mouseClicked(MouseEvent): void
mouseEntered(MouseEvent): void
mouseExited(MouseEvent): void
mousePressed(MouseEvent): void
mouseReleased(MouseEvent): void
mouseDragged(MouseEvent): void
mouseMoved(MouseEvent): void
processMouseEvents(MouseEvent): void
addCloseableTabbedPaneListener(ClosableTabbedPaneListener):
removeCloseableTabbedPaneListener(ClosableTabbedPaneListen
getCloseableTabbedPaneListener(): ClosableTabbedPaneListener
fireCloseTab(int): boolean
<<Java Class>>
HypAuthEditCutActio
hauthor
HypAuthEditCutAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
HypAuthEditCopySpecialActio
hauthor
HypAuthEditCopySpecialAction()
actionPerformed(ActionEvent): void
update(): void
getProperties(): void
<<Java Class>>
CopyDialog
hauthor
copyAll: JRadioButton
copyStructure: JRadioButton
helpTopicId: String
COPY_STRUCTURE: ir
COPY_ALL: int
CopyDialog(Frame,String)
getCopyMode(): int
<<Java Class>>
CloseableMetalTabbedPaneUI
hauthor
horizontalTextPosition: int
CloseableMetalTabbedPaneUI(
CloseableMetalTabbedPaneUI(int
layoutLabel(int,FontMetrics,int,String,Icon,Rectangle,Rectangle,Rectangle,boolean): void
horizontalTe
CloseableT
CloseableT
layoutLabel

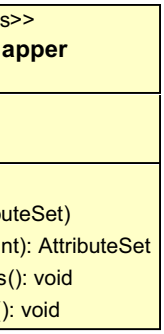

Cell
Cell
oid
Edge): void
: String
<<Java Class>>
Encoder
hauthor
graphDocuments: Hashtable
diagrama: Document
sourceURL: URL
MAIN_SEQ_CLOSED_MARK: Strin
IMAGE_DIR: String
Encoder(ControladorGrafo)
addGraph(Document,GraphInfo): void
isFirstNode(Node): boolean
setCurrent(GraphInfo): void
openGraph(GraphInfo): Object[]
getContentsFromFile(File): String
changeDTDDeclaration(String,String): String
setContentsInFile(File,String): void
graphToArray(Document): Object[]
getNode(int,Object[]): DefaultGraphCell
getAttributes(Node,Properties): Properties
saveGraph(Object[],GraphInfo): Document
addNode(Object,HyperGraphCell): Nodo
addBounds(Element,DefaultGraphCell): void
addAllLinks(Element,Nodo,HyperGraphCell): void
addLink(Element,String,String): void
zipFile(String): void
unzipFile(String,String): void
-encoder
0..1
#encoder
0..1
<<Java Interface>>
ClosableTabbedPaneListener
hauthor
closeTab(int): boolean
<<Java Class>>
CloseTabIcon
hauthor
x_pos: int
y_pos: int
width: int
height: int
fileIcon: Icon
mouseover: boolean
mousepressed: boolean
CloseTabIcon(Icon)
paintIcon(Component,Graphics,int,int): void
getIconWidth(): int
getIconHeight(): int
getBounds(): Rectangle
void
er): void
[]
<<Java Class>>
HyperErrorHandler
hauthor
res: boolean
filename: String
fatalError(SAXParseException): void
error(SAXParseException): void
warning(SAXParseException): void
<<Java Class>>
CloseableTabbedPaneUI
hauthor
extPosition: int
abbedPaneUI()
abbedPaneUI(int)
(int,FontMetrics,int,String,Icon,Rectangle,Rectangle,Rectangle,boolean): void
<<Java Class>>
GraphNameMissingException
hauthor
GraphNameMissingException()
GraphNameMissingException(String)

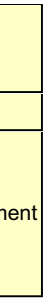

oneEdgeSelected(): boolean
edgesSelected(): boolean
allSelected(): boolean
allNodesSelected(): boolean
saveInProgress(): boolean
-graphManager
0..1
crearSecLineal(int,double,double,Color): DefaultGraphCell[]
añadirSecLineal(int,int,int,int): void
añadirBucleAbierto(HyperGraphCell,int,double,double,boolean): DefaultGraphCell[]
añadirSatelite(HyperGraphCell,int): void
alargarSecuencia(boolean): void
crearEnlace(DefaultGraphCell,DefaultGraphCell,String): void
borrarEnlace(String,Hashtable,HyperGraphCell): void
~graphManager
0..1
markAsCut(): void
markAsNotCut(): void
deleteSelection(): void
checkCutAborted(): void
getVertices(Object[]): Object[]
getEdges(Object[]): Object[]
isVertex(Object): boolean
isGroup(Object): boolean
nodeInList(Object[],Object): boolean
areConsecutive(Object[]): boolean
isLoopComplete(Object[]): boolean
isStructureComplete(Object[]): boolean
containsLoops(Object[]): boolean
getLoops(Object[]): Object[]
getLoopOrigins(Object[]): Object[]
deleteEdge(DefaultGraphCell,DefaultGraphCell): void
valueChanged(GraphSelectionEvent): void
undo(): void
canUndo(): boolean
redo(): void
canRedo(): boolean
startCompoundAction(): void
endCompoundAction(): void
copyFromGraph(ActionEvent): void
pasteToGraph(ActionEvent,boolean): void
canPaste(): boolean
isValidSelection(): boolean
processInsertion(Object[]): Object[]
replaceInsertion(Object[],boolean): void
replaceLoops(Object[],Object[]): void
adjustConnections(DefaultGraphCell,DefaultGraphCell,DefaultGraphCell,DefaultGraphCell): Vector
changeIds(Object[]): Hashtable
resetUserObjects(Object[]): void
resetLoopEdges(Object[]): void
adaptUserObjects(Object[]): void
copyNodo(Nodo): Nodo
saveGraph(): void
getSequence(Object): Object[]
saveGraphDocument(): Document
resumeGraph(): Object[]
createTempDir(): URL
createImageDir(): void
deleteTempDir(): void
addGraphPaneListener(GraphPaneListener): void
fireNameChanged(): void

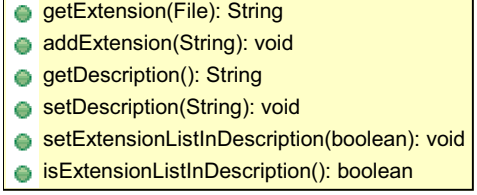
HyperViewer
Planos UML

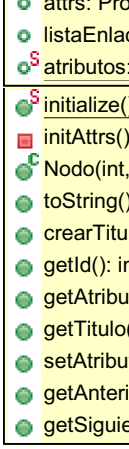

<<Java Class>>
HyperViewer
hviewer
mainFrame: Frame
resources: ResourceBundle
APP_NAME: String
APP_TEMP_DIR: Strin
IMAGE_DIR: String
appTempDir: File
g: Graphics
panelPrincipal: JPanel
panelCentral: JPanel
panelIzquierdo: JPanel
panelDerecho: JPanel
panelSur: JPanel
panelNorte: JPanel
colorFondo: Color
scrollPane: JScrollPane
mi_historia: JList
sourceURL: String
HyperViewer()
initAppTempDir(): void
getAppTempDir(): File
createImageDir(): void
customizeFrame(): void
processWindowEvent(WindowEvent): void
showOpenFileDialog(): void
visualizeNode(Nodo): void
getNodo(String): Nodo
resetScrollBar(): void
addElementTreePanel(): void
main(String[]): void
<<Java Class>>
DynamicResource
hviewer
DynamicResource()
getResourceString(ResourceBundle,String): String
getResource(ResourceBundle,String): URL
+dynRes
0..1
~decoder
~gui
0..1
0..1
source: String
tempDir: URL
diagrama: Do
mainSeqClos
INCORRECT
DEFAULT_G
MAIN_SEQ_
HyperDecode
openGraph(S
unzipFile(Stri
getContentsF
changeDTDD
setContentsIn
graphToArray
closeMainSeq
getUserObjec
getAttributes(
getListaEnlac
getFirstNode(
getLastNode(
<<Java Class>>
OpenFileDialog
hviewer
fileName: JTextField
browseFile: JButton
buttonPanel: JPanel
okButton: JButton
cancelButton: JButton
ignoreActions: boolean
imgDir: File
imgFile: String
helpTopicId: String
result: int
RESULT_CANCEL: in
RESULT_OK: in
FILE_LAST_OPEN: String
OpenFileDialog(Frame,String)
buildButtonGroup(): void
setIgnoreActions(boolean): void
getPathFromUrl(URL,String): String
chooseFile(): File
setFileName(String): void
getFileName(): String
actionPerformed(ActionEvent): void
cancel(): void
confirm(): void
getResult(): int
+elementTreePane
0..1
<<Java Class>>
ElementTreePanel
hviewer
tree: JTree
bookName: String
updatingSelection: boolean
MyHistoryNode: DefaultMutableTreeNode
ElementTreePanel(String,HyperEditorPane
updateMyHistory(Nodo): void
valueChanged(TreeSelectionEvent): void
getTree(): JTree
getTreeModel(): DefaultTreeModel
#treeModel
0..1
<<Java Class>>
HyperTreeModel
hviewer
bookName: String
HyperTreeModel(String)
<<Java Class>>
MyFileFilter
hviewer
TYPE_UNKNOWN: String
HIDDEN_FILE: Strin
filters: Hashtable
description: String
fullDescription: String
useExtensionsInDescription: boolean
MyFileFilter()
MyFileFilter(String)
MyFileFilter(String,String)
MyFileFilter(String[])
MyFileFilter(String[],String)
accept(File): boolean
INICIO_
PAGINA:
FIN_CAR
tipos: Str
count: int
id: int
type: int

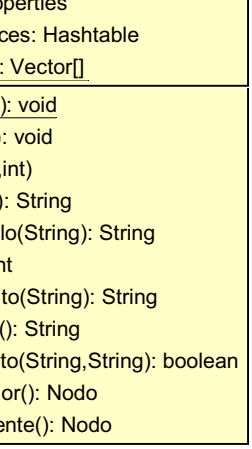

<<Java Class>>
HyperErrorHandler
hviewer
res: boolean
filename: String
fatalError(SAXParseException): void
error(SAXParseException): void
warning(SAXParseException): void
<<Java Class>>
HyperDecoder
hviewer
ed: boolean
_LINK_MARK: Strin
RAPH_FILE: Strin
CLOSED_MARK: Strin
er(HyperViewer)
tring): Object[]
ng,String): void
romFile(File): String
eclaration(String,String): String
nFile(File,String): void
(Document): Object[]
q(): void
ct(Node): Nodo
(Node,Properties): Properties
es(Node): Hashtable
): Nodo
): Nodo
<<Java Class>>
Datos
hviewer
libro: Object[]
Datos(Object[])
getNodoByID(String): Nodo
+datos
0..1
<<Java Class>>
HyperEditorPane
hviewer
image: Image
enlacesActivos: boolean
clickOnEnlace: boolean
HyperEditorPane()
paintComponent(Graphics): void
followLink(HyperlinkEvent): void
display(Nodo): void
desactivarEnlaces(): void
activarEnlaces(): void
getMaxAttributes(Element,StyleSheet): AttributeSet
applyAttributes(AttributeSet,Iterator,boolean,boolean): void
~areaTexto
0..1
#editor
0..1
<<Java Clas
AttributeM
hviewer
toHTML: int
toJava: int
AttributeMapper()
AttributeMapper(Attrib
getMappedAttributes(
mapToHTMLAttribute
mapToJavaAttributes(
<<Java Class>>
HyperEditorKit
hviewer
defaultFactory: ViewFactory
HyperEditorKit()
HyperEditorKit(String)
createDefaultDocument(): Docun
read(Reader,Document,int): void
getViewFactory(): ViewFactory
<<Java Class>>
HyperEditorListener
hviewer
origen: int
fin: int
margen: int
mouseClicked(MouseEvent): void
<<Java Class>>
SHTMLFactory
hviewer
create(Element): View
<<Java Class>>
Nodo
hviewer
CAPITULO: i
: int
PITULO: ir
ing[]
t[]
~nodo
0..1
<<Java Class>>
InvisibleView
hviewer
isVisible: boolean
InvisibleView(Element)
getPreferredSpan(int): float

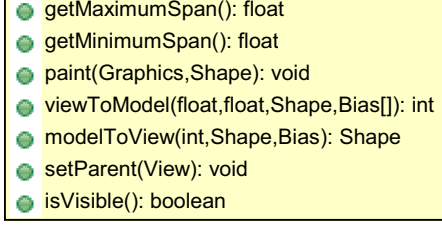

s>>
apper
uteSet)
nt): AttributeSet
s(): void
): void


nent


getExtension(File): String
addExtension(String): void
getDescription(): String
setDescription(String): void
setExtensionListInDescription(boolean): void
isExtensionListInDescription(): boolean


attrs: Pro
listaEnlac
atributos:
initialize(
initAttrs()
Nodo(int,
toString()
crearTitu
getId(): in
getAtribu
getTitulo
setAtribu
getAnteri
getSiguie


pperties
ces: Hashtable
: Vector[]
): void
: void
,int)
): String
lo(String): String
nt
to(String): String
(): String
to(String,String): boolean
or(): Nodo
ente(): Nodo


getMaximumSpan(): float
getMinimumSpan(): float
paint(Graphics,Shape): void
viewToModel(float,float,Shape,Bias[]): int
modelToView(int,Shape,Bias): Shape
setParent(View): void
isVisible(): boolean

Pliego III
Manuales de usuario

HyperAuthor
Manual de usuario

Contenido

1. Sobre HyperAuthor

HyperAuthor es una herramienta que permite crear gráficamente estructuras hipertextuales tomando como base, de entre todas las posibles estructuras tipo básicas de las que se componen la mayoría de los hipertextos, aquellas que permiten construir hiperficciones de calidad con una estructura que no provoque en el lector sensaciones de pérdida o desorientación. Así mismo permite también la inclusión de contenido en la estructura, creando de esta manera una obra hipertextual completa.

Con un diseño similar al de un editor de texto sencillo, HyperAuthor permite construir diagramas de forma dinámica y cómoda para el usuario. Las estructuras hipertextuales básicas se representan como objetos gráficos que pueden ser insertados en el panel de dibujo sin más que hacer clic sobre el botón adecuado de la barra de herramientas. Una vez allí, los nodos que la componen pueden ser editados, copiados, cortados, eliminados…

A semejanza de cualquier otro editor, HyperAuthor puede guardar en disco los diagramas con los que está operando así como abrir aquellos diagramas previamente guardados. Una vez haya creado su estructura hipertextual utilizando HyperAuthor, usted podrá dotarla de contenido introduciendo texto (con formato o sin él) así como imágenes en cada uno de los nodos que la componen. Para realizar esta tarea, HyperAuthor incorpora un editor de nodos: un editor de texto similar a aquellos con los que ya está familiarizado al que se le han añadido algunas funciones adicionales específicas de la escritura de hiperficción.

HyperAuthor es una aplicación especialmente diseñada para personas pertenecientes al mundo de las Humanidades, a quienes no se les presupone un gran conocimiento tecnológico, que quieren iniciarse en la literatura hipertextual. Pretende ser una herramienta interactiva y divertida que le permita a usted, usuario, comprobar "in situ" cómo se puede construir una red hipertextual a partir de unas estructuras tipo básicas, cómo se puede dotar de contenido a esta red construyendo con ello una hiperficción coherente y completa y familiarizarse con las situaciones para las que cada una de estas estructuras básicas es más adecuada.

Para ayudarle en esta tarea, a lo largo de este manual encontrará algunas notas marcadas con el siguiente símbolo  . Estas notas le explicarán los conceptos y nociones básicas que no están directamente relacionado con HyperAuthor sino con las estructuras o técnicas usadas en la creación de literatura hipertextual y que, probablemente, le ayuden a fijar sus ideas. Si usted está familiarizado con el empleo de las estructuras básicas hipertextuales así como con la creación de hiperficciones, puede saltar estas notas sin miedo alguno.

Además de éstas, a lo largo del texto del manual encontrará otras notas marcadas con el símbolo  que le aclararán aspectos de tipo práctico (auxiliares) que puede que ya conozca, y que por lo tanto, sólo necesitará consultar si tiene una duda concreta.

2. Empezando

A estas alturas ya ha leído un poco sobre esta herramienta, lo que hace y sobre cómo y para qué debe usarse. Es, por lo tanto, un buen momento para echarle un vistazo general. Para empezar con buen pie deberá, por supuesto, iniciar la aplicación.

Iniciando HyperAuthor

HyperAuthor se proporciona en dos formatos diferentes, cada uno de los cuales tiene su propia forma de inicialización:

- *Ejecutable (.exe). Sólo para sistemas Windows. Para iniciar la aplicación basta con hacer doble clic sobre el archivo .exe*

- *Archivo jar. Válido tanto en sistemas Windows como Linux. La aplicación puede iniciarse de dos formas:*

  a) *En línea de comandos mediante la instrucción "java –jar HyperAuthor.java".*

  b) *Haciendo doble clic sobre el archivo .jar si dispone de una versión de java mayor o igual a 1.6*
  (usted puede comprobar este punto tecleando "java –version" en su línea de comandos).

En la siguiente figura se muestra la vista principal de la aplicación. Esta ventana aparecerá en su pantalla nada más iniciarla (a excepción de la pestaña en la que ya se han introducido algunos nodos):

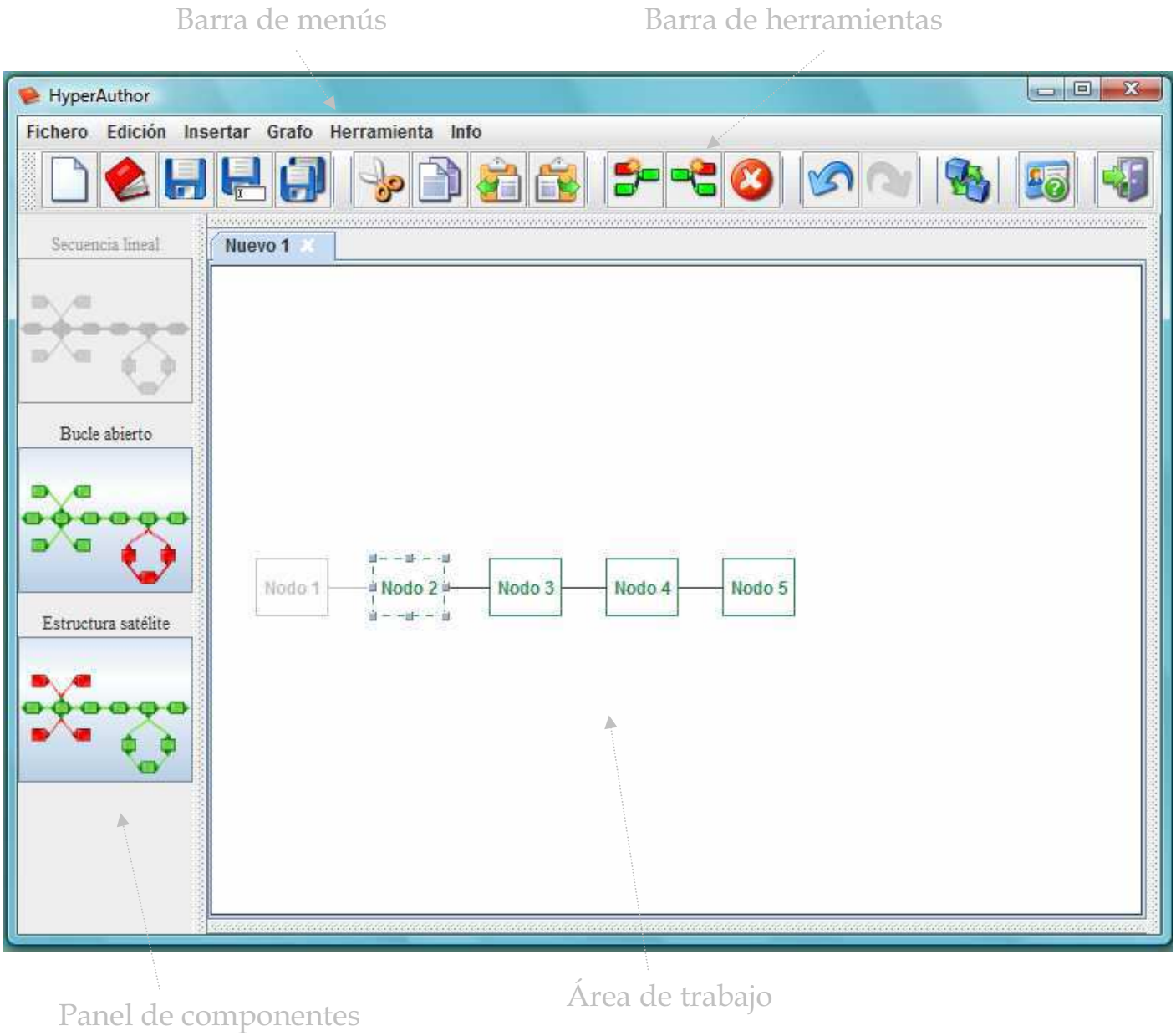


Figura 1. Ventana principal de HyperAuthor.

Como se puede observar con un simple vistazo, HyperAuthor, al igual que la gran mayoría de las aplicaciones actuales, dispone de una barra de menús así como de una barra de herramientas que contiene las acciones principales que se pueden realizar con la aplicación.

Los menús contenidos en la barra son básicamente los propios de un editor al uso (véase Archivo, Edición, Info) a los que se suman tres de funcionalidad específica (Insertar, Grafo y Herramienta). La utilidad de cada uno de estos menús se describirá más adelante.

En la barra de herramientas los vistosos botones que la componen y que se pueden observar en la figura proporcionan un acceso rápido a las acciones más frecuentes. Si pasa el ratón sobre cualquiera de ellos y lo mantiene durante unos segundos, aparecerá una pequeña descripción de la acción que se lleva a cabo tras pulsar sobre él.

En la parte izquierda de la ventana de la aplicación se encuentra la zona de control. En ella se sitúa el panel de componentes que contiene los elementos que puede utilizar para crear su red hipertextual (o lo que es lo mismo, los ladrillos con los que va a construir su casa). Estos componentes son las estructuras tipo básicas, aquéllas estructuras a partir de las cuales y mediante su combinación, se puede crear cualquier hipertexto.

Estructuras hipertextuales básicas

Son los diferentes tipos de estructura de los que se componen las redes hipertextuales creadas en HyperAuthor. Todo hipertexto bien diseñado y organizado se compone de muchas de estas estructuras, combinadas de forma más o menos compleja. Estas estructuras no son universales, es decir, se pueden encontrar obras hipertextuales que no contengan ninguna de estas estructuras. Sin embargo, su utilización como unidad mínima de construcción garantiza que el hipertexto resultante no sea desconcertante ni complicado de navegar para el lector.

Para añadir una de estas estructuras al diagrama que está construyendo tan sólo deberá hacer clic sobre una de ellas. También puede insertar una estructura mediante el menú Insertar de la barra de menús.

Aviso

Los botones de inserción de secuencias permanecerán deshabilitados a no ser que tenga algún diagrama abierto sobre el que colocarlas. Cuando hay diagramas abiertos en la aplicación, éstos aparecen en pestañas sobre el área de trabajo situado en la zona central, tal y como se ve en la Figura 1.

En el centro de la ventana, ocupando el espacio mayor se encuentra el área de dibujo donde, como ya hemos dicho, se muestran en diferentes pestañas los diagramas abiertos en HyperAuthor en un determinado momento. En esta área aparecerán las estructuras insertadas que componen cada diagrama, y se podrán seleccionar, arrastrar, copiar, cortar, pegar, borrar, etc. dentro del mismo diagrama o entre diagramas distintos.

Hasta este momento se ha realizado un breve recorrido a vista de pájaro por la aplicación. Este recorrido le habrá resultado de ayuda para entrar en contacto con la herramienta; aún así, sería buena idea que antes de comenzar la lectura de los siguientes

apartados se familiarizase un poco con ella explorando las opciones ofrecidas por los botones de la barra de herramientas, las opciones del menú, etc.

Los puntos restantes en este apartado contienen detalles de la operación con HyperAuthor (sus nombres son muy indicativos a ese respecto), y puesto que estará deseando entrar en dinámica, pasemos a ellos sin más dilación.

2.1 La primera hiperficción

Por supuesto, la creación de un diagrama es la primera y más fundamental de las operaciones a realizar con HyperAuthor, ya que el objetivo de esta aplicación no es otro que el que usted pueda crear sus propios hipertextos.

Comenzaremos por la creación de un diagrama nuevo. Para ello, vaya al menú Archivo->Nuevo o pulse el botón nuevo () en la barra de herramientas.

Nota

Si realiza las operaciones desde la barra de menús, observará que a la derecha de la mayoría de los comandos aparece una combinación de letras. Esa abreviatura le permitirá ejecutar el comando directamente desde el teclado.

En el área de dibujo aparecerá una pestaña llamada "Nuevo" que contendrá el diagrama que se dispone a crear. La aplicación le preguntará en este momento por la longitud en nodos que desea que tenga la secuencia lineal principal del hipertexto.

Es importante tener en cuenta que todo diagrama creado por HyperAuthor debe contener al menos una secuencia lineal "pura", que actuará como su secuencia principal (posteriormente se verá que esta secuencia principal puede convertirse en un bucle abierto, usando una funcionalidad que permite enlazar el nodo final con el inicial, pero eso será más adelante).

Esta secuencia debe ser siempre la primera en insertarse en el área de trabajo, y por este motivo la herramienta la inserta de forma automática cada vez que se abre un diagrama nuevo. De hecho, observará que cuando el área de dibujo se encuentre vacía, el único botón de inserción de estructuras que se encuentra habilitado es aquel correspondiente a la secuencia lineal.

Formato básico de un hipertexto

Todo hipertexto creado en HyperAuthor debe contener una estructura principal de forma lineal. Esta estructura llevará el peso del relato y será la encargada de hacer avanzar la acción. Este requisito tiene como objetivo que las obras creadas con esta aplicación no disten demasiado de la literatura tradicional, de forma que no resulten chocantes a los lectores. Sin embargo, la aplicación le ofrece la posibilidad de enlazar el nodo final con el primero, de manera que la secuencia principal se convierta en un bucle abierto; esto no la excluye de la obligatoriedad de ser la primera en ser introducida.

Una vez disponga de una pestaña en la que se ha insertado la secuencia principal, puede comenzar a dibujar su diagrama en ella añadiendo estructuras y combinándolas de la manera que le resulte más adecuada.

Para añadir una estructura seleccione, en primer lugar, un nodo de referencia de entre los ya introducidos. Este nodo será el origen y fin de la nueva secuencia. Al seleccionar un nodo, los botones de inserción correspondientes a secuencias de tipo bucle abierto y satélite se habilitarán. Pulse sobre el botón adecuado y aparecerá un cuadro de diálogo en el que deberá introducir el número de nodos que desea que tenga la nueva secuencia. Una vez haya introducido el número deseado, pulse "Aceptar". La nueva estructura aparecerá en el área de dibujo integrada mediante las conexiones oportunas con el nodo origen, aquel que usted marcó como referencia.

Este proceso de adición de nuevas estructuras puede llevarse también acabo desde el menú emergente asociado al diagrama. Para utilizar esta forma de insertar secuencias, seleccione, al igual que con el método anterior, un nodo de referencia. Una vez seleccionado, haga clic sobre el con el botón secundario del ratón. En este momento aparecerá un menú emergente entre cuyas opciones se encuentra la de añadir bucles abiertos o estructuras tipo satélite. Seleccione la que le parezca conveniente y aparecerá, al igual que en el caso anterior, un cuadro de diálogo en el que debe introducir el número de nodos que desea que contenga la nueva secuencia. Pulse "Aceptar" una vez introducido el número de nodos y la nueva secuencia aparecerá en el área de trabajo.

Pero vamos a ilustrar todo esto con un sencillo ejemplo. Crearemos un programa que contenga una secuencia lineal principal y un bucle abierto conectado a ella. Con este pequeño ejemplo se mostrará el funcionamiento básico de la herramienta.

En primer lugar, al abrir una nueva pestaña la herramienta introducirá de forma semiautomática la secuencia lineal principal; simplemente debe escoger su longitud en nodos y la herramienta hará el resto del trabajo. En este caso escogemos un número reducido de nodos, con cinco será suficiente para que el ejemplo sea lo suficientemente ilustrativo. Tras pulsar "Aceptar", en el área de trabajo tendremos lo que aparece en la siguiente figura.

Como puede observar, los nodos en la nueva secuencia están unidos mediante enlaces "secuenciales", lo que implica que la lectura puede ir en ambos sentidos. Este tipo de enlace forma parte de todas las secuencias que usted tiene disponibles en la herramienta, e indica un movimiento del tipo "Anterior" o "Siguiente" dentro de las mismas.

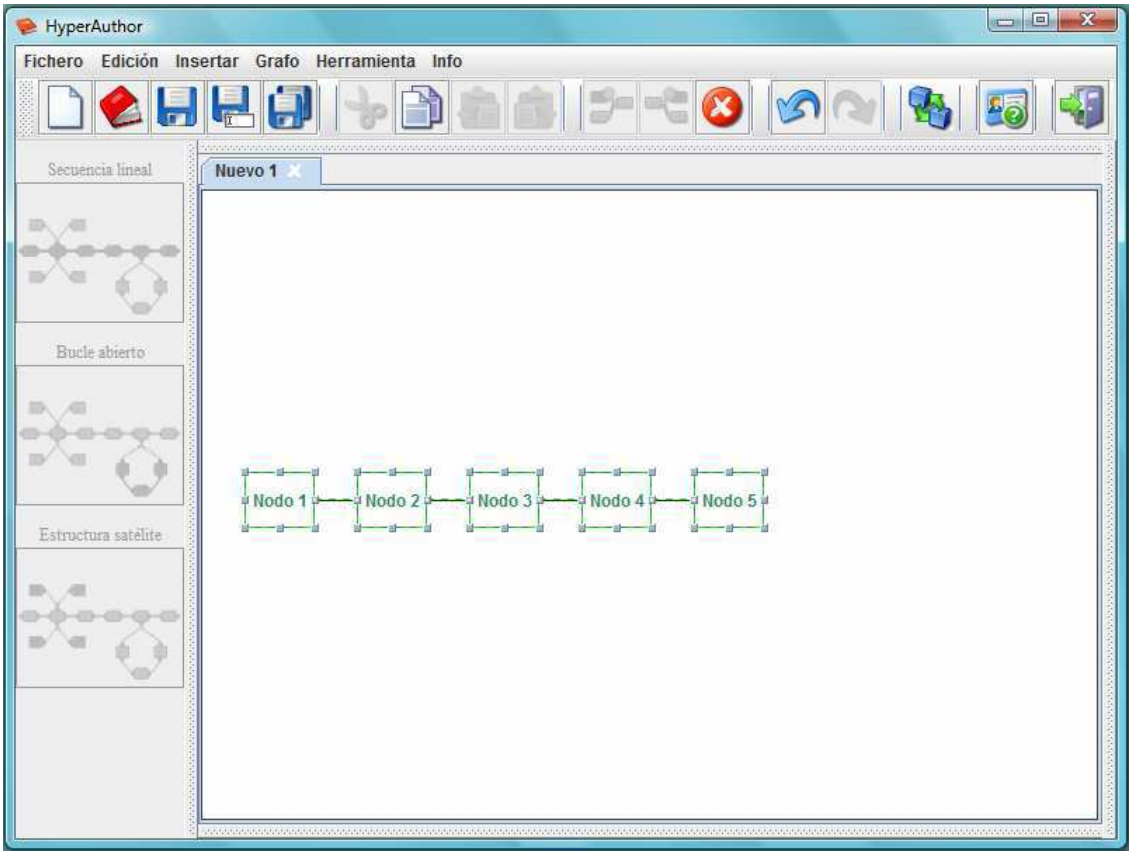


Figura 2. Vista de HyperAuthor tras insertar la secuencia lineal principal.

El siguiente paso a realizar en este momento es añadir a la red hipertextual, por ejemplo, un bucle abierto. Esta estructura no puede añadirse de forma independiente, sino que debe hacerlo conectada a la estructura principal que ya hemos introducido. Para esto, vamos a seguir los siguientes pasos.

1. Seleccione el nodo de referencia que será el nodo origen y fin de la nueva secuencia. En este caso se ha escogido el nodo 2. Una vez seleccionado, se habilitan los botones de inserción de secuencias de la barra izquierda.
2. Haga clic sobre el botón correspondiente al tipo de secuencia que desea insertar, en el caso que nos ocupa, el de insertar un bucle abierto.
3. Indique, en el diálogo que aparecerá tras pulsar el botón, el número de nodos que va a contener la nueva secuencia. Nuestro nuevo bucle abierto estará formado por tres nodos.

Tras seguir estos tres pasos, la nueva secuencia queda insertada e integrada junto con el resto de la estructura existente tal y como se puede observar en la figura que aparece a continuación.

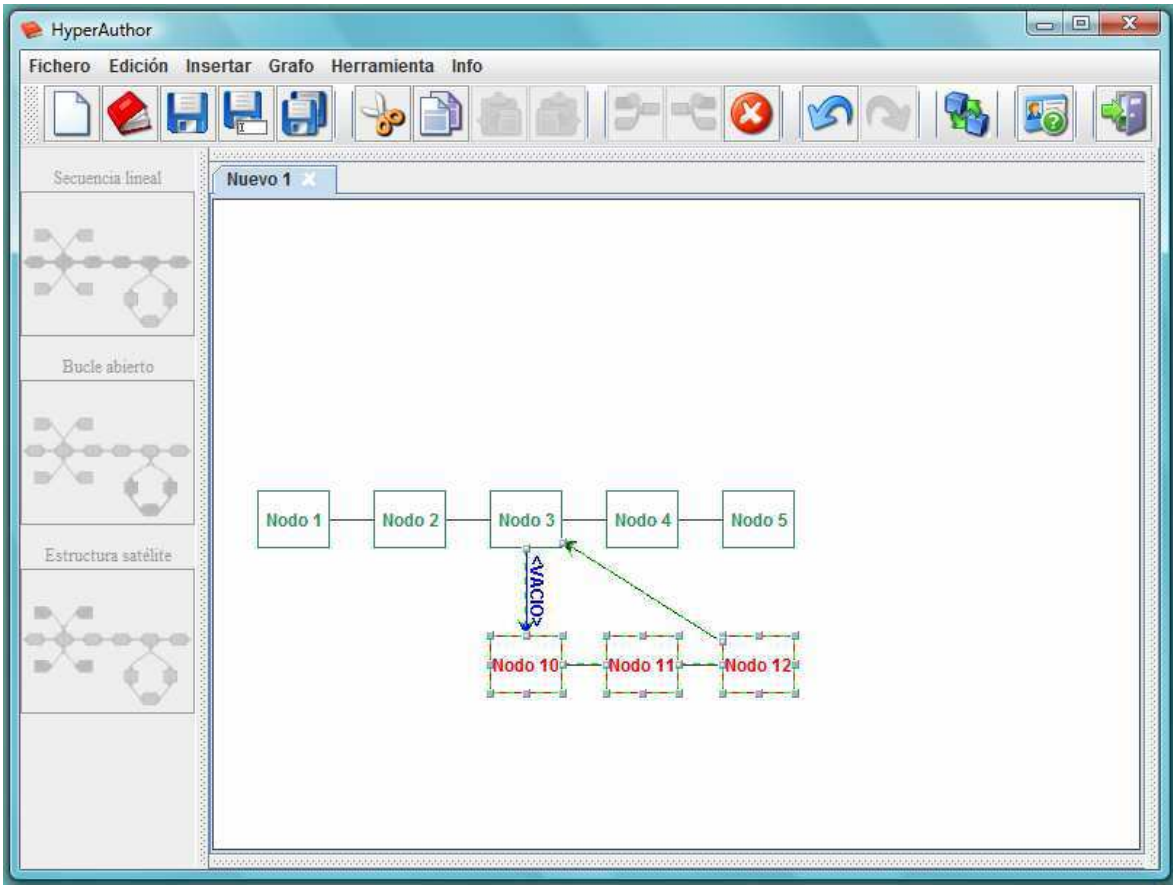


Figura 3. HyperAuthor tras insertar la secuencia lineal principal y un bucle abierto.

En este caso se puede apreciar cómo la introducción del bucle abierto ha dado lugar a la creación, de manera automática, de las conexiones entre él y la secuencia principal. Al utilizar este tipo de estructuras que rompen la linealidad del relato, la herramienta introduce un nuevo tipo de enlaces, los enlaces de bifurcación. Este nuevo tipo de enlace se representa mediante flechas y su dirección de apuntamiento indica el flujo de lectura al tomar ese camino.

Reglas para la interconexión de estructuras

Existen una serie de reglas que HyperAuthor fuerza a respetar a la hora de insertar una nueva secuencia e integrarla con la estructura correspondiente. Si una de estas

reglas no se cumple, la herramienta no dará opción a la inserción de la nueva secuencia. Estas reglas son las siguientes:

1. *Un enlace de bifurcación tiene como origen un único nodo.*
2. *Un enlace de bifurcación tiene como destino un único nodo.*
3. *Un único nodo puede ser origen de varios enlaces de bifurcación.*
4. *Un nodo puede ser destino de un único enlace de bifurcación*

Note, de la misma manera, que aquél enlace en el que recae realmente la ruptura de la linealidad es aquél que aparece marcado como "<VACIO>". Esta marca indica que aún este enlace no tiene una palabra asociada en el texto del nodo. Asociar estos enlaces con una palabra contenida en el texto del propio nodo es fundamental para que este camino de lectura pueda ser seguido por el futuro lector de la obra; de otra manera, nunca estará disponible para él.

Caminos de lectura en un hipertexto

Un hipertexto no es más que un conjunto de nodos cuya estructura ofrece al lector diferentes posibles caminos de lectura. Las posibles bifurcaciones en la secuencia lineal pueden ser muy simples o muy complejas, pero en los hipertextos creados con HyperAuthor devolverán siempre al lector al punto de partida.

El trabajo del autor consiste en construir una estructura compuesta por varios caminos diferentes dispuestos de forma que la lectura resulte una experiencia satisfactoria. El lector recorrerá los caminos que escoja recorriendo las secuencias en un orden determinado y totalmente secuencial. En ocasiones recorrerá un determinado camino y en otras otro, en función de sus propias decisiones de navegación.

Estas decisiones de navegación se ofrecen al lector en forma de enlaces. Los enlaces son palabras contenidas en el texto de la obra que aparecerán marcadas de una forma diferente al resto. Cuando están activados, el lector simplemente necesita hacer clic sobre la palabra marcada como enlace para que un nuevo camino de lectura se abra ante él.

Por este motivo, toda bifurcación en la estructura hipertextual debe estar asociada a una determinada palabra en el texto. De esta manera, la palabra en cuestión aparecerá marcada como enlace y el usuario entenderá que encierra un nuevo trayecto. De otra forma, ese trayecto quedará oculto para el usuario, ya que no existirá palabra marcada como enlace asociada al mismo y el usuario desconocerá su existencia.

Una vez que se han integrado las dos estructuras que componen nuestro pequeño ejemplo (la secuencia principal y el bucle abierto) es necesario un último ajuste antes de dar por terminada esta mini obra: dotar de contenido a los nodos que la componen.

Para acceder al editor de nodos, mediante el cual podrá dotar de contenido al nodo que desee, basta con hacer doble clic sobre el nodo con el que se desea trabajar. El aspecto del editor de nodos es similar al de cualquier editor de texto al uso, por lo que su manejo le resultará sencillo.

Tanto el contenido del nodo como el título del mismo pueden ser modificados desde el editor. Para dotar al nodo de contenido basta con emplear el área de trabajo destinada a tal efecto; ésta se sitúa en el área central de la ventana y ocupa la mayor parte del espacio disponible. Introduzca el texto deseado en el área de trabajo.

En cuanto al título, habrá podido observar que los nodos son creados con un título por defecto. Este título consiste en la palabra "Nodo" seguida del número de orden en creación que ocupa el nodo en un determinado hipertexto. Para modificar este título, pulse en el menú "Archivo" y haga clic sobre "Título del nodo". En la siguiente figura se muestra la interfaz del editor de nodos así como la ventana de edición que aparecerá tras realizar estos pasos.

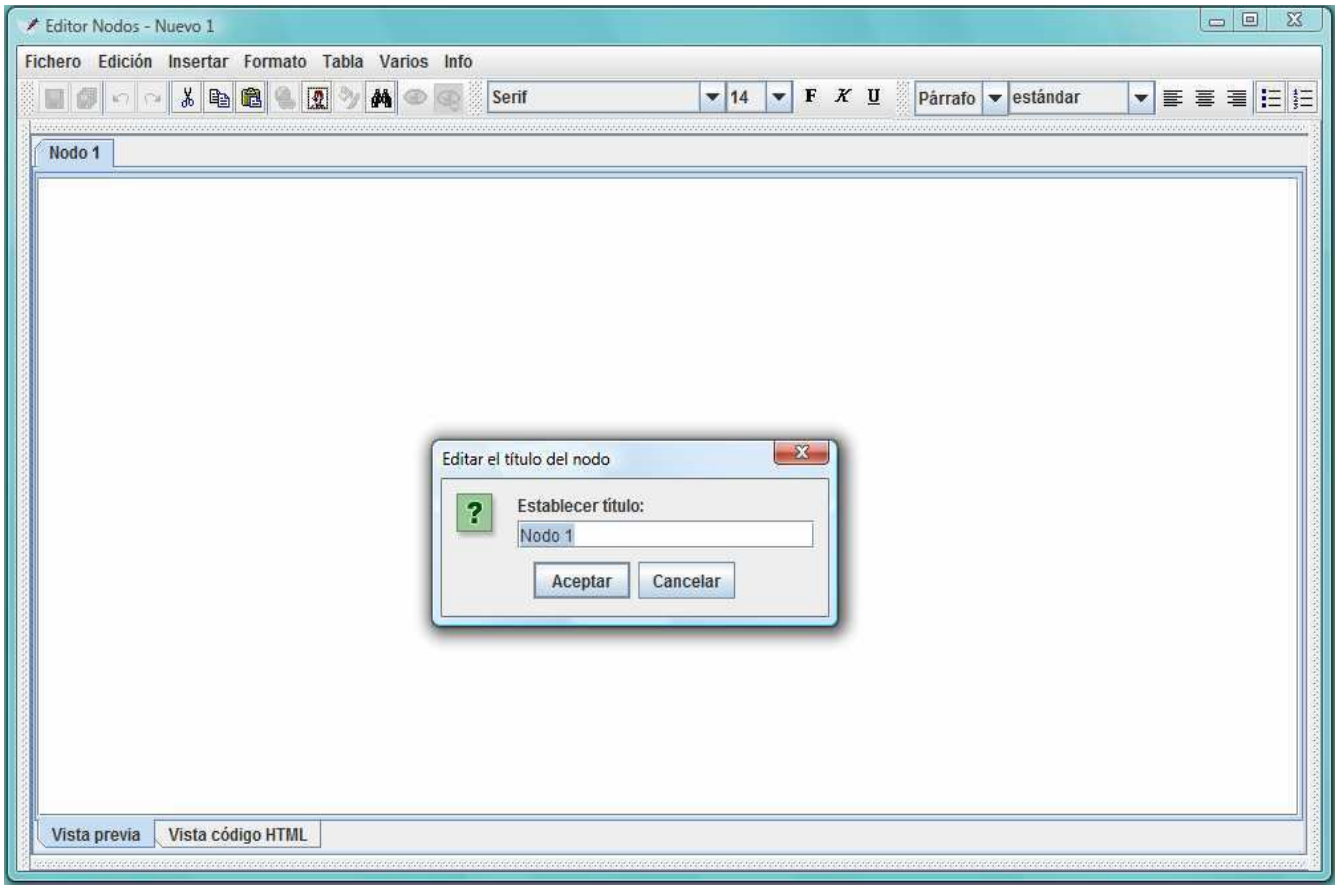


Figura 4. Vista del editor de nodos empleado en HyperAuthor abierto con un nodo en blanco y cuadro de diálogo donde se modifica el título del nodo abierto.

Introduzca el texto en el campo habilitado para tal efecto y pulse Aceptar. Pulse el botón de guardar en la barra de herramientas. Ya ha editado su primer nodo.

Nota

Es necesario que, tras editar un nodo, pulse sobre el botón de guardar de la barra de herramientas del editor para que los cambios introducidos queden registrados y se apliquen a la estructura de forma que ésta se actualice de forma adecuada.

Repita el proceso de edición con todos los nodos de los que se compone la obra siguiendo los pasos anteriormente mencionados. Para tener la obra completa simplemente resta por hacer un último paso. Como se comentó anteriormente, los enlaces de bifurcación necesitan tener su correspondiente texto en el contenido del nodo origen para que el usuario pueda activar ese camino de lectura al navegar por la obra. En nuestra obra existe un único enlace de bifurcación, aquél que conecta el bucle abierto con la secuencia principal. Vamos, por lo tanto, a crear dentro del texto del nodo origen del bucle abierto el enlace asociado necesario.

Para llevar a cabo esta última tarea debe seguir los siguientes pasos:

1. Abra el nodo en cuestión en el editor para su edición.
2. Introduzca en el área de escritura el texto que desee que contenga el nodo.
3. Seleccione la palabra escogida para actuar de enlace. Si el nodo con el que está trabajando es origen de enlaces de bifurcación que aún no tienen palabra asociada en el texto, el botón de edición de enlaces situado en la barra de herramientas se habilitará. En caso contrario, el botón permanecerá deshabilitado.

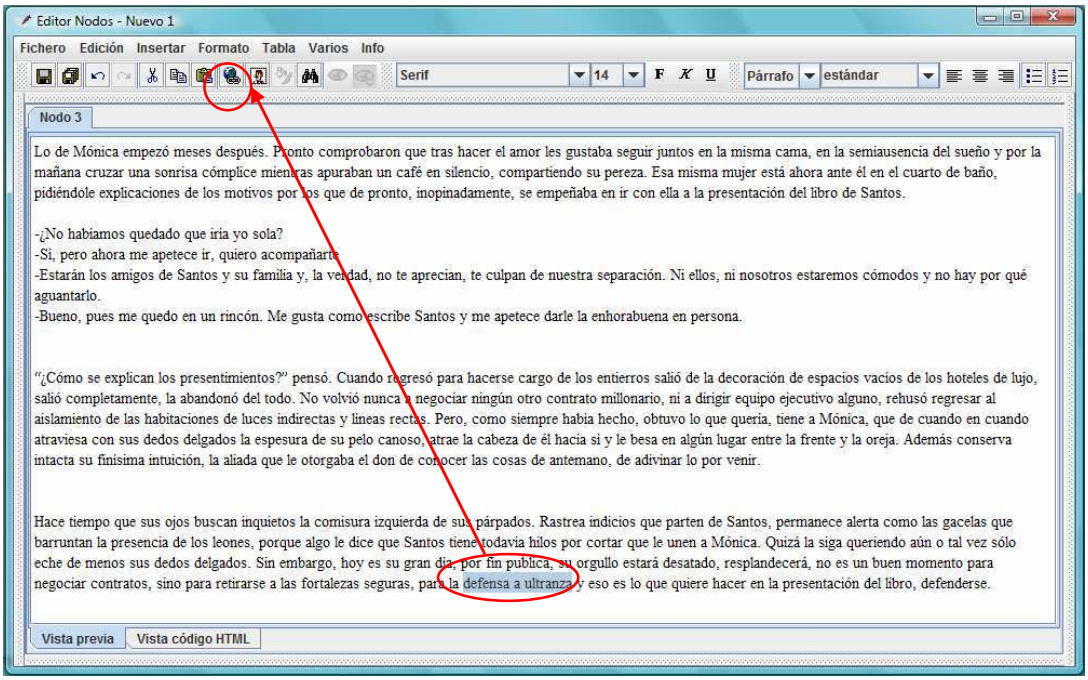


Figura 5. Creación de enlaces en HyperAuthor.

4. Una vez que el botón de edición de enlaces se haya habilitado, haga clic sobre él. Aparecerá la siguiente ventana de diálogo. En ella debe seleccionar el nodo destino del enlace (lo que permitirá diferenciar este enlace de otros posibles enlaces de bifurcación existentes en el mismo nodo) así como el estilo con el que desea que sea marcado. Por defecto los enlaces se marcan en azul y con subrayado. En esta ventana puede también modificar el texto escogido para el enlace. Si lo hace, el nuevo texto reemplazará al texto seleccionado en el área de texto.

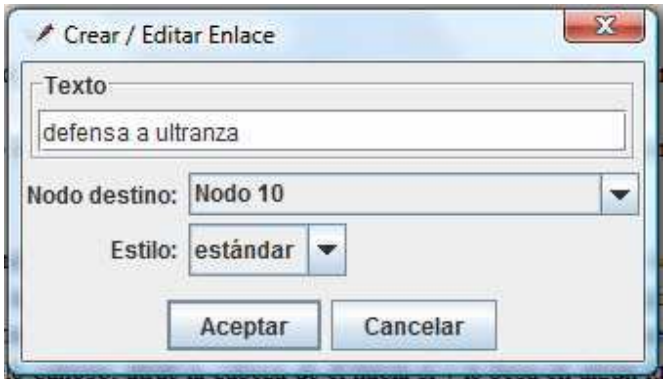


Figura 6. Cuadro de dialogo destinado a la creación y edición de enlaces en HyperAuthor.

5. Una vez haya concluido con la edición del enlace, pulse Aceptar. Ya ha creado su enlace.

Y, ¡voilá!, acaba de crear también su primera obra hipertextual en HyperAuthor.

2.2 Creación y edición de un diagrama

Como se ha comentado en los primeros párrafos de este apartado, HyperAuthor, de forma similar a cualquier otro editor, puede almacenar las hiperficciones que se crean en él y posteriormente abrir aquellas hiperficciones que se hayan almacenado en disco.

En el punto anterior hemos visto cómo crear un diagrama desde cero. Para abrir un diagrama existente simplemente diríjase al menú Archivo -> Abrir, o bien pulse el botón abrir () en la barra de herramientas.

Una vez esté el diagrama en su pestaña correspondiente, usted dispone para modificarlo o para ayudarle a crearlo, de una serie de facilidades gráficas asociadas con los nodos y las estructuras. Entre estas funcionalidades se encuentran, como es lógico, las de insertar bucle abierto o secuencia tipo satélite, pero también existen otras como son arrastrar, copiar, cortar y pegar en el mismo diagrama o entre diagramas diferentes, eliminar nodos de una determinada secuencia para acortar su longitud o añadirlos para

aumentarla, etc. Todas estas funcionalidades están distribuidas entre el menú Edición y el menú Insertar de la barra de menús. Están también accesibles de forma redundante mediante los botones de la barra de herramientas así como desde el menú emergente que aparece al pulsar el botón derecho del ratón sobre un nodo.

Todas estas funcionalidades pueden aplicarse tanto a nodos sueltos como a un grupo de nodos que formen parte de la misma estructura (en ningún caso a enlaces de forma aislada).

Para seleccionar un único nodo, tan sólo haga clic con el ratón sobre el mismo. Aparecerá un borde y unos pequeños cuadrados blancos alrededor del nodo que le indica que el elemento está seleccionado. Una vez seleccionado, un nodo puede redimensionarse sin más que pulsar y arrastrar de estos cuadrados.

Para seleccionar un grupo de elementos, pulse sobre cada uno de ellos manteniendo pulsada la tecla Ctrl, o bien trace con el ratón un rectángulo que los contenga haciendo clic en un punto cualquiera de la pantalla y arrastrando el ratón en diagonal.

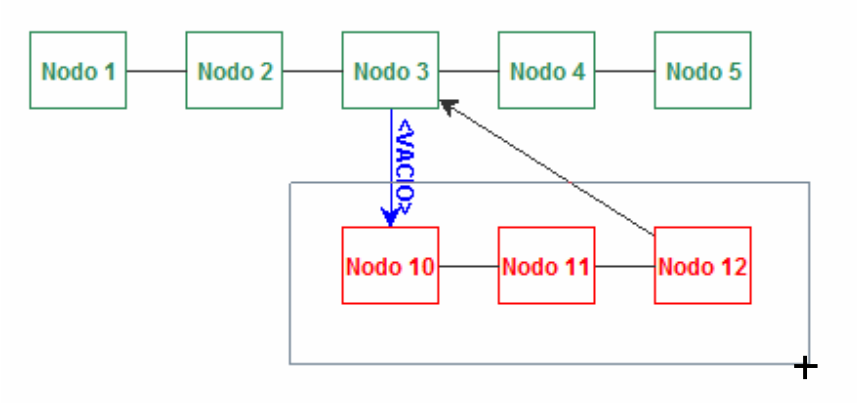


Figura 7. Selección múltiple de nodos en HyperAuthor.

Durante la utilización de las funcionalidades mencionadas en los párrafos anteriores la aplicación comprueba, antes de llevarlas a cabo, que la estructura hipertextual resultante de la realización de las mismas será correcta. No es posible, por ejemplo, suprimir un nodo de forma aislada si de él parte un bucle abierto o una estructura tipo satélite, ya que tras su eliminación estas estructuras quedarían "huérfanas", sin nodo origen. Por lo tanto, no debe extrañarse si al intentar llevar a cabo una operación de cortado, recibe mensajes de error que le impiden llevar esta operación a cabo.

Corrección de una red hipertextual

Con la finalidad de que el resultado obtenido usando HyperAuthor sean redes hipertextuales bien diseñadas que resulten atractivas para el lector, la herramienta

impone una serie de reglas en cuanto a la estructura que el diagrama con el que trabaja debe cumplir en todo momento.

Es por este motivo que, antes de llevar a cabo cualquiera de las funcionalidades de edición disponibles, la herramienta revisa si la acción provocará que el diagrama permanezca en un estado de validez o no. Si esto no es así, dará error y no permitirá realizar la acción en cuestión.

No se permitirán, por lo tanto, acciones que tengan como resultado dejar nodos sueltos sin conexión con el resto de la red, estructuras "huérfanas" (que no tengan nodo origen) o incluso secuencias de bifurcación que tengan un nodo final distinto al de origen.

En el apartado 4 encontrará todos los detalles sobre cada una de las funciones de edición e inserción y su modo de uso.

Una vez haya construido el diagrama usando todas aquellas funcionalidades que haya considerado oportunas y lo haya dotado de contenido en la manera que se explicó en el aparatado 2.1, podrá almacenarlo en disco para poder utilizarlo en cualquier otro momento. Para ello dispone de las operaciones clásicas de "Guardar" y "Guardar como", disponibles tanto en la barra de menús (bajo el menú "Archivo") como en la barra de herramientas con los botones guardar () y guardar como().

Existe una tercera opción para el almacenamiento de diagramas en disco, y ésta es la operación "Guardar todos", accesible de nuevo tanto bajo el menú "Archivo" como desde la barra de herramientas a través del botón guardar todo (). Esta opción es de utilidad cuando se está trabajando con varios diagramas de manera simultánea en la herramienta y se quieren guardar los cambios en todos ellos, ya que al accionar esta función la herramienta guardará todos los hipertextos abiertos en ese momento uno a uno. Al igual que sucede si se utiliza la opción "Guardar", la herramienta guardará automáticamente los cambios en aquellos diagramas que ya tengan un nombre asignado y preguntará por el nombre que se le quiere asignar a aquellos que sea la primera vez que van a ser salvados.

HyperAuthor almacena los diagramas en archivos XML con el formato especificados en la DTD "htxt.dtd". Como además se permite la inclusión de imágenes y texto con formato en cada uno de los nodos que conforman la obra, junto a este fichero XML se genera también una carpeta que contiene una copia de las imágenes empleadas y una CSS en la que se almacenan los estilos de texto creados por el usuario. Estos tres elementos se almacenan en una única carpeta que se comprime y que es la que tendrá el nombre y se ubicará en la ruta definida por el usuario.

XML, DTD y CSS

XML es simplemente un lenguaje de marcado, como el popular HTML, y de la misma forma que éste, está basado en etiquetas. Las etiquetas que se pueden usar en cada archivo, así como el formato de éstas – atributos, hijos, etc.- están especificadas y definidas en un archivo DTD, o archivo de definición de datos.

Por su parte, la CSS hace lo propio con los atributos de presentación en pantalla. En XML existen etiquetas que hacen referencia al estilo del texto que se va a mostrar en pantalla. Para poder separar el propio contenido del archivo de su formato, la CSS es un fichero en el que se almacenan todos esos datos de formato para cada estilo que se quiere que esté presente en el fichero XML. De esta forma, basta con referenciar el estilo deseado desde el fichero y éste tendrá el formato adecuado en su representación en pantalla.

2.3 Enlaces en HyperAuthor

Enlaces

Un enlace no es más que la representación de un camino de bifurcación dentro de la estructura hipertextual. De manera general, los enlaces se representan como una porción de texto marcada de manera especial dentro de un determinado documento. Los enlaces se caracterizan por actuar como nexo de unión entre un documento origen y otro destino (o dos fragmentos distintos dentro del mismo documento).

Se puede pensar en un enlace como un puntero a un lugar distinto al que se encuentra. De manera habitual, origen y destino contienen información relacionada. De esta manera el usuario puede acceder a la información del destino en el momento en el que lo desee.

Cada enlace está representado por una porción de texto, normalmente denominada nombre, con la cual se le reconoce.

Los enlaces pueden ser de distinto tipo dependiendo del tipo de relación que establezca entre origen y destino. Puede haber enlaces que creen jerarquías o enlaces que comuniquen información a un mismo nivel. También existen enlaces interno o externos dependiendo de si el destino está fuera o dentro del documento en el que está contenido, etc.

En los diagramas que puede construir en HyperAuthor se pueden utilizar dos tipos diferentes de enlaces: los enlaces secuenciales y los enlaces de bifurcación.

El primer tipo de enlace debe su nombre al hecho de que se utilizan durante la navegación en el interior de estructuras secuenciales. Su principal característica es que todos ellos son bidireccionales, es decir, el flujo de lectura se puede dar en los dos posibles sentidos. Este tipo de enlaces permite al lector una navegación del tipo anterior o siguiente, y es, con diferencia, el tipo de enlace más abundante.

Los enlaces secuenciales se encuentran presentes tanto en las estructuras lineales como en los bucles abiertos ya que, en estos últimos, una vez tomado el camino de lectura el lector se encuentra dentro de una secuencia lineal con la única peculiaridad de que esta secuencia le devolverá al nodo del que partió. Los enlaces secuenciales unen, por lo tanto, todos los nodos que componen un bucle abierto así como aquellos que forman parte de la secuencia lineal principal.

El segundo tipo de enlace es aquel que implica una ruptura en la linealidad de la lectura. Estos enlaces se encuentran presentes de forma mayoritaria en las secuencias tipo satélite, pero también en las conexiones de los bucles abiertos a la secuencia cuya linealidad fracturan. Los enlaces de bifurcación ofrecen al usuario la opción de navegar por los diferentes trayectos de los que está compuesto el hipertexto; por este motivo, para permitir que el usuario entre en estos caminos disponibles, este tipo de enlaces deben estar unidos a un texto en el contenido del nodo. Cuando no lo están, se marcan como <VACÍO>.

Para construir algo diferente a una estructura lineal es necesario que trabaje con este último tipo de enlaces. Sin embargo, deberá tener en cuenta las siguientes características de los mismos cuando opere con ellos.

a. Todos los enlaces en HyperAuthor conectan nodos.
b. Todo enlace creado en el contenido de un nodo debe tener su correspondencia en la estructura. Es decir, sólo se podrán crear enlaces en aquellos nodos que son origen de un enlace de bifurcación marcado como <VACIO>.
c. Todos los enlaces están identificados por su nombre, que no tiene por qué ser único dentro de la hiperficción, ni siquiera dentro del nodo.
d. En cualquier momento se puede cambiar el destino de cualquiera de los enlaces siempre que exista otro enlace vacío en la estructura hipertextual que tenga un destino diferente. En este caso el nuevo destino se puede elegir de entre los destinos de todos los enlaces marcados como vacíos en la estructura que tengan como origen el nodo en el que se encuentra.

3. Estructuras básicas

Uno de los mayores riesgos de la literatura hipertextual consiste en la creación de redes hipertextuales demasiado complejas, sin ninguna estructura aparente, en las que el lector pronto se siente perdido. Para evitar este problema, HyperAuthor basa el proceso de creación de la estructura del hipertexto en la inserción y combinación de estructuras como unidad mínima (y no de nodos de manera independiente).

Cada una de las estructuras básicas disponibles en HyperAuthor ha sido cuidadosamente escogida con el fin de que la creación de hiperficciones resulte lo más sencilla e intuitiva posible, consiguiendo al mismo tiempo buenos resultados.

Estructura ó secuencia.

Una estructura ó secuencia es, en esencia, un conjunto de nodos unidos mediante enlaces secuenciales (del tipo anterior y siguiente) con una forma determinada. El número de estructuras diferentes que se pueden encontrar en los hipertextos actuales es muy elevado y la complejidad varía de unas a otras.

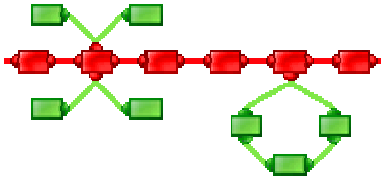
Existen estructuras simples y compuestas. Compuestas son aquellas que se pueden descomponer en estructuras simples. En HyperAuthor se incluyen como secuencias básicas aquellas estructuras simples a partir de las cuales, mediante combinación, se puede construir el resto de estructuras existentes.

Cada una de estas secuencias básicas tiene un propósito y una funcionalidad concretas, determinados en gran medida por su forma estructural.

A continuación se describen cada una de las estructuras disponibles en la herramienta así como su funcionalidad y las propiedades que las caracterizan.

Como se ha comentado con anterioridad, para editar los nodos que componen cada una de las estructuras y dotarlas así de contenido, basta con hacer doble clic sobre uno de ellos. Para cambiar el nodo que está editando, no es necesario que cierre el editor y abra uno nuevo. Basta con que haga doble clic sobre el nuevo nodo que quiera editar o que cambie de pestaña en el editor si ya lo tiene abierto.

3.1 Secuencia lineal



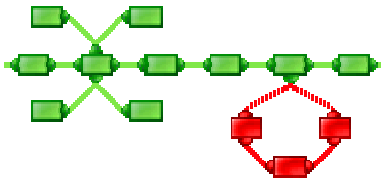
Esta secuencia representa el tipo de estructura en el que se sustenta la literatura tradicional. Es el tipo de estructura que conforma la secuencia principal de lectura en el hipertexto y con ella se definen un principio y un final para la obra, aunque más adelante se contará como esta estructura principal lineal puede convertirse en un bucle abierto.

Nota sobre su uso

Se utiliza para hacer avanzar el relato y es además indispensable para evitar que el usuario se vea forzado a activar constantemente enlaces para navegar durante la lectura, lo que resultaría agotador.

Este tipo de estructura es de uso obligatorio y debe ser la primera en introducirse en el área de dibujo. Tomando esta secuencia como base se creará el resto de la estructura hipertextual.

3.2 Bucle abierto



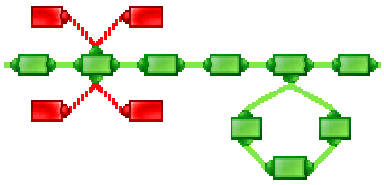
Representa una ruptura en la linealidad del relato, ofreciendo un camino de lectura diferente a aquel que se venía siguiendo hasta el momento.

Se trata de una secuencia lineal cuyo origen y destino es un único nodo ubicado en la secuencia de la que ramifica, de modo que cuando se ha pasado por todos los nodos que la componen, se vuelve al nodo del relato central del que se partió.

Nota sobre su uso

Este tipo de estructura fue propuesto por de las Heras en su libro "Navegar por la información" como el único tipo de estructura necesaria para crear un hipertexto bien diseñado. Son especialmente útiles para insertar digresiones, líneas argumentales secundarias, información adicional sobre un personaje concreto, etc. Implican, básicamente, una ampliación de lo narrado.

3.3 Secuencia satélite



Este tipo de secuencia, al igual que el bucle abierto, representa también una ruptura en la linealidad del relato. De hecho, a diferencia de éste que ofrece una única ramificación, una estructura tipo satélite implica varias ramificaciones "simultáneas" partiendo de un mismo nodo.

En este caso se trata de una estructura formada por un nodo central del que parten diferentes ramas, formadas también cada una de ellas por un único nodo, y que tienen como origen y destino el nodo central.

Nota sobre su uso

Aunque en principio este tipo de secuencia pudiera no parecer de especial relevancia, es en realidad útil hasta el punto que una de las primeras obras hipertextuales y de mayor importancia creada hasta el momento la usa en exclusiva en su estructura. Se trata de "Afternoon, a story", un clásico de la narrativa hipertextual.

El uso de este tipo de estructura es interesante, por ejemplo, cuando se quiere presentar un mismo hecho desde el punto de vista de diferentes personajes o cuando se pretende hacer confluir diferentes caminos narrativos hacia un mismo hecho común.

En este punto habrá probablemente notado que los dos tipos de secuencias ofrecidos para romper la linealidad del relato terminan en el mismo punto del que partieron. Esto, obviamente, no es casualidad. De esta forma el usuario de las obras que cree con HyperAuthor no tendrá la sensación de "estar perdiéndose algo importante" que normalmente se experimenta al elegir un camino de lectura y descartar el resto.

El lector volverá siempre al punto de partida y, en ese momento, será libre de escoger si quiere explorar algún otro de los caminos disponibles o si por el contrario desea avanzar en el relato. Se evita, con esto, que un enlace provoque la anulación del resto.

Secuencias alternativas

Sin duda, este es un tipo de secuencia muy conocido y empleado. Se trata de una secuencia cuyo uso principal consiste en el relato de experiencias diferentes y excluyentes que podrían ocurrir de forma paralela. Puede ser también utilizada para narrar un hecho centrado en cómo lo vive un determinado personaje o para seguir en un momento puntual un lugar, un hecho, etc. De forma resumida se puede decir que se usa cuando el deseo de profundizar en algo debe, al mismo tiempo, hacer avanzar la historia.

A pesar de su aparente utilidad, este tipo de estructura se ha quedado deliberadamente fuera del ámbito de HyperAuthor debido a que la ramificación de la narración no termina en el punto donde empezó, con lo que se corre el riesgo comentado anteriormente de que el usuario sienta insatisfacción y frustración por tener que dejar caminos sin explorar en su lectura.

4. Edición de un diagrama

Como ya se ha comentado con anterioridad, HyperAuthor ofrece todo un conjunto de funcionalidades de edición destinadas a facilitar el proceso de creación de un diagrama. Estas funcionalidades son todas aquellas que se encuentran distribuidas entre los menús "Edición" e "Insertar", y en este apartado encontrará todos los detalles sobre cada una de ellas así como su modo de uso.

4.1 Deshacer y rehacer

Esta funcionalidad se encuentra accesible desde el menú Edición y de forma redundante desde los botones marcados como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, estas tareas tienen asociadas una combinación de teclas que permiten usarlas directamente desde el teclado. Esta combinación es Ctrl+Z para deshacer y Ctrl + Y para rehacer.

El nombre de estas tareas es auto explicativo, ya que se rehace o deshace la última acción llevada a cabo sobre el diagrama. Como acción se entiende cualquier inserción de estructuras, aumento o disminución de la longitud de las ya existentes, reubicación de nodos, cortado, copiado, pegado y supresión de nodos.

No se pueden deshacer o rehacer aquellas acciones llevadas a cabo en el editor de nodos, ya que para eso existen las funciones de deshacer y rehacer propias del mismo. Tampoco se incluyen entre estas acciones aquellas que se llevan a cabo sobre el grafo como consecuencia de una acción realizada en el editor de nodos. Por ejemplo, si se crea un enlace en un determinado nodo, el enlace de bifurcación correspondiente en la estructura dejará de estar marcado como vacío. Este cambio no se puede deshacer mediante esta funcionalidad ya que no ha sido llevado a cabo directamente por el usuario sobre el área de trabajo.

4.2 Cortar

Esta funcionalidad se encuentra accesible desde el menú Edición y de forma redundante desde el botón marcado como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, esta tarea tiene asociada una combinación de teclas que permite usarla directamente desde el teclado. En este caso la combinación es Ctrl+X.

Esta opción permite cortar tanto nodos sueltos como estructuras completas para pegarlas posteriormente en otro sito del diagrama o incluso en otro diagrama. Para cortar uno o varios elementos, simplemente es necesario seleccionar mediante el uso del ratón los elementos que desee cortar y pulsar posteriormente sobre el botón de la barra de herramientas destinado a tal efecto.

Antes de hacer efectivo el cortado de los elementos seleccionados, la herramienta realiza un proceso de comprobación sobre el resultado de la acción para asegurarse de que

el diagrama seguirá siendo válido después de esta operación (ver *"Corrección de una red hipertextual" en el apartado 2.2).*

Condiciones

Como regla general las operaciones de cortado, copiado o supresión deben cumplir los siguientes requisitos:

- *Se puede cortar, copiar o suprimir un nodo suelto siempre que este no sea origen de alguna bifurcación.*
- *Se puede cortar, copiar o suprimir un nodo junto con una bifurcación si esa bifurcación es única y se corta, copia o suprime de forma completa.*
- *Se puede cortar, copiar o pegar un nodo origen de varias bifurcaciones si junto con el se cortan, copia o suprimen cada una de sus ramificaciones de forma completa.*

Si los requisitos necesarios para llevar a cabo una operación de este estilo no se cumplen, en pantalla aparecerá el siguiente error:

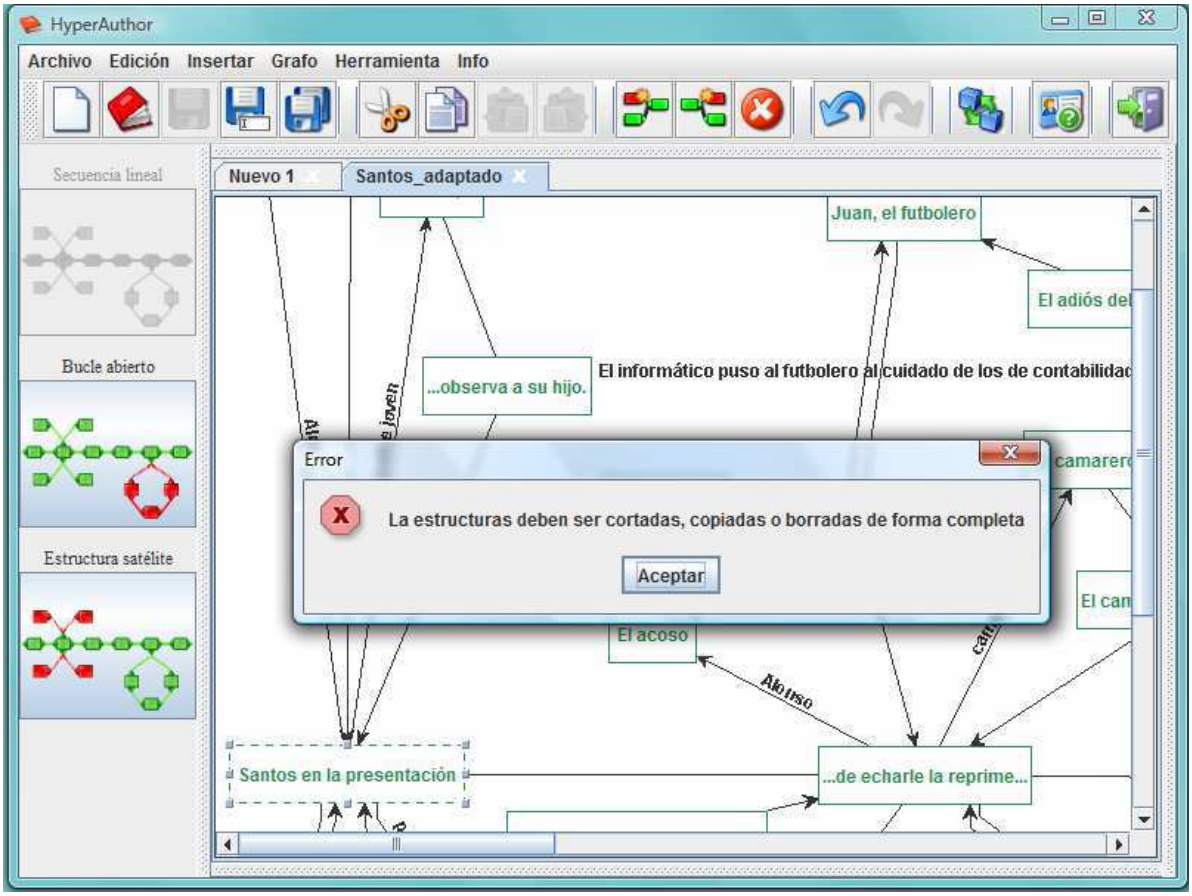


Figura 8. Mensaje de error en HyperAuthor.

Si, por el contrario, la selección cumple con todas las condiciones expuestas, la operación puede llevarse a cabo.

Nota

El resultado de una operación de cortado no dará en ningún caso lugar a la supresión de los elementos cortados. Una vez la operación de cortado ha sido invocada, los elementos correspondientes se marcan como cortados dibujándolos de un color gris claro. Estos nodos así marcados no se pueden utilizar para realizar ninguna otra operación sobre ellos. En esta situación hay dos posibles opciones:

- *Si de forma inmediatamente posterior al cortado se lleva a cabo una operación de pegado, los nodos marcados como cortados desaparecen de su ubicación original y pasan a la ubicación que usted le indique.*
- *Si de forma inmediatamente posterior al cortado se lleva a cabo cualquier otra operación diferente a la de pegado, la herramienta entiende que la operación de cortado se ha abortado y en ese momento los elementos que antes estaban marcados como cortados dejan de estarlo.*

4.3 Copiar y Copiar especial

Estas funcionalidades se encuentran accesibles desde el menú Edición y de forma redundante desde los botones marcados como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, la función “Copiar” tiene asociada una combinación de teclas que permite usarla directamente desde el teclado. En este caso la combinación es Ctrl+C. La función “Copiar especial” no tiene teclas de acceso rápido asociadas.

Estas opciones permiten copiar tanto nodos sueltos como estructuras completas para pegarlas posteriormente en otro sito del diagrama o incluso en otro diagrama. Para copiar uno o varios elementos, simplemente es necesario seleccionar mediante el uso del ratón los elementos que desee cortar y pulsar posteriormente sobre el botón de la barra de herramientas destinado a tal efecto.

Antes de hacer efectivo el copiado de los elementos seleccionados, la herramienta realiza un proceso de comprobación sobre el resultado de la acción para asegurarse de que

el diagrama seguirá siendo válido después de esta operación (ver "*Corrección de una red hipertextual" en el apartado 2.2)*.

Si la selección de elementos sobre la que se va a llevar a cabo la operación de copiado no es adecuada, en pantalla aparecerá un mensaje de error similar al que se muestra en el punto 4.2. En este caso, la operación de copiado no se llevará a cabo (ver *"Condiciones"* en el apartado 4.2).

Hasta ahora todo lo comentado en este apartado es válido para cualquiera de las dos funciones que aquí se tratan. Sin embargo, estas dos funcionalidades no son idénticas. La función "Copiar" y "Copiar especial" se diferencian en lo siguiente:

"Copiar" trabaja tanto con la estructura de los nodos copiados como con su contenido, lo que significa que cuando se invoca esta función se realiza una copia de cada uno de los elementos seleccionados con el mismo contenido que tenían los originales (a excepción de los destinos de los enlaces que la herramienta modificará de forma automática para ajustarse a la ubicación de los nuevos nodos).

Sin embargo, la función "Copiar especial" permite elegir si se quiere copiar todo, en cuyo caso su función es exacta a la de "Copiar", o si por el contrario se quiere copiar sólo la estructura de los elementos seleccionados. De esta forma, cuando usted haga clic sobre el botón asociado a "Copiar especial" en la barra de herramientas, aparecerá la siguiente ventana en la que usted deberá escoger con qué quiere trabajar.

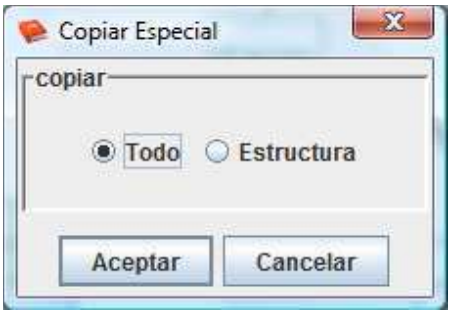


Figura 9. Opciones que permite la funcionalidad "Copiar Especial".

La opción "Todo" es la que viene marcada por defecto, y como se ha explicado, esta operación es equivalente a la de "Copiar". Si por el contrario selecciona "Estructura", sólo se copiará una versión en blanco de los nodos seleccionados. Esta última opción se muestra especialmente útil cuando se pretende replicar en otra parte de la red de hipertexto una estructura más o menos compleja a la que se quiere dotar de contenido diferente al de la original.

4.4 Pegar izquierda y Pegar derecha

Estas funcionalidades se encuentran accesibles desde el menú Edición y de forma redundante desde los botones marcados como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, la función "Pegar izquierda" tiene asociada una combinación de teclas que permite usarla directamente desde el teclado. En este caso la combinación es Ctrl+V. La función "Pegar derecha" no tiene teclas de acceso rápido asociadas.

Estas opciones permiten pegar aquellos nodos o estructuras completas que hayan sido previamente cortadas o copiadas de cualquiera de las dos formas posibles y que, por lo tanto, se encuentren en el *clipboard* del sistema.

Como usted habrá podido comprobar, las opciones "Pegar izquierda" o "Pegar derecha" necesitan de una referencia a la izquierda o a la derecha de la cual integrar los nodos que van a ser pegados. Pues bien, para pegar el contenido del *clipboard* usted deberá, en primer lugar, seleccionar el nodo de referencia para la operación. Para ello basta con hacer clic con el ratón sobre el nodo que desee. Las acciones de pegado no se habilitarán hasta que usted no haya seleccionado el nodo de referencia.

Una vez escogido y seleccionado el mencionado nodo de referencia, basta con que pulse sobre cualquiera de los botones destinados a las acciones de pegado para que ésta se lleve a cabo. La herramienta creará las conexiones necesarias para que los nuevos nodos queden perfectamente integrados en la red de hipertexto.

Si la operación de pegado ha sido invocada de manera inmediatamente posterior a una de cortado, no sólo se llevará a cabo la pertinente integración de los nodos correspondientes en el lugar seleccionado por usted, sino que también se borrarán los nodos que hasta el momento aparecían marcados como cortados.

4.5 Borrar

Esta funcionalidad se encuentra accesible desde el menú Edición y de forma redundante desde el botón marcado como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, esta tarea tiene asociada una combinación de teclas que permite usarla directamente desde el teclado. En este caso la combinación es SUPR.

Esta opción permite suprimir tanto nodos sueltos como estructuras completas para hacerlas desaparecer definitivamente del diagrama. Para borrar uno o varios elementos, simplemente es necesario seleccionar mediante el uso del ratón los elementos que desee borrar y pulsar posteriormente sobre el botón de la barra de herramientas destinado a tal efecto.

Antes de hacer efectivo el borrado de los elementos seleccionados, la herramienta realiza un proceso de comprobación sobre el resultado de la acción para asegurarse de que el diagrama seguirá siendo válido después de esta operación (ver "*Corrección de una red hipertextual" en el apartado 2.2)*.

Si la selección de elementos sobre la que se va a llevar a cabo la operación de borrado no es adecuada, en pantalla aparecerá un mensaje de error similar al que se muestra en el punto 4.2. En este caso, la operación de borrado no se llevará a cabo (ver *"Condiciones"* en el apartado 4.2).

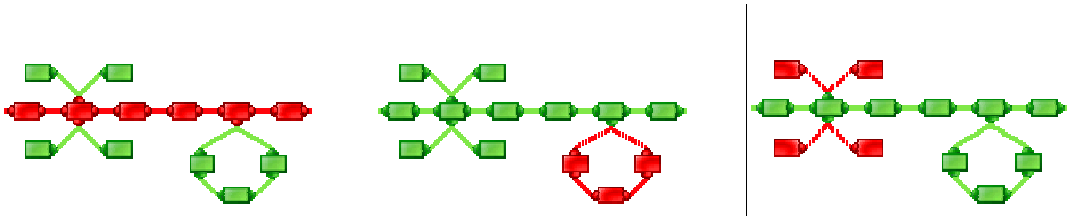
4.6 Seleccionar todo

Esta funcionalidad se encuentra accesible desde el menú Edición y de forma redundante desde el botón marcado como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el panel de trabajo con el botón secundario del ratón.

Para facilitar el uso y la accesibilidad de la herramienta, esta tarea tiene asociada una combinación de teclas que permite usarla directamente desde el teclado. En este caso la combinación es Ctrl+A.

Esta opción permite seleccionar todos los elementos que haya insertado hasta el momento en el área de dibujo sin necesidad de tener que ir seleccionándolos uno a uno.

4.7 Insertar secuencia lineal, bucle abierto o estructura satélite



Estas funcionalidades se encuentran accesibles desde el menú Insertar y de forma redundante desde los botones especiales situados en el panel de componentes que aparece en la parte izquierda de la ventana de la aplicación y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Estas funciones, debido que tienen un panel propio para facilitar el acceso a ellas, no tienen asociada ninguna combinación de teclas de acceso rápido.

Para una descripción detalla del uso de estas estructuras, por favor, remítase al apartado 2.1 La primera hiperficción.

4.8 Insertar nuevo nodo a derecha o izquierda

Estas funcionalidades se encuentran accesibles desde el menú Insertar y de forma redundante desde los botones marcados como se indica en la barra de herramientas y en el menú emergente que aparece al hacer clic sobre el grafo con el botón secundario del ratón.

Estas funciones, al igual que el resto de las tareas de inserción, no disponen de una combinación de teclas de acceso rápido asociadas.

Estas funciones permiten añadir un nodo a una determinada secuencia ya insertada en el área de trabajo con el fin de alargar la longitud de la misma sin tener que borrar la secuencia existente e introducir en su lugar una con una longitud mayor.

Al igual que ocurría en el caso de las funciones de pegado (ver apartado 4.4), las opciones “Insertar nuevo nodo a la izquierda” o “Insertar nuevo nodo a la derecha” necesitan de una referencia a la izquierda o a la derecha de la cual insertar el nodo que va a ser introducido.

Pues bien, para insertar un nodo nuevo usted deberá, en primer lugar, seleccionar el nodo de referencia para la operación. Para ello basta con hacer clic con el ratón sobre el

nodo que desee. Estas funciones relacionadas con la inserción de un nuevo nodo no se habilitarán hasta que no usted no haya seleccionado el nodo de referencia.

Una vez escogido y seleccionado el mencionado nodo de referencia, basta con que pulse sobre cualquiera de los botones destinados a las acciones de inserción para que ésta se lleve a cabo. La herramienta creará las conexiones necesarias para que los nuevos nodos queden perfectamente integrados en la red de hipertexto.

4.9 Abrir/Cerrar bucle en secuencia principal

Esta funcionalidad es la única de todas las mencionadas que se halla fuera de los menús de Edición e Insertar. Ésta opción se encuentra accesible desde el menú Grafo y de forma redundante desde los botones marcados como se indica en la barra de herramientas.

Esta función permite convertir la secuencia lineal en la que se basa la estructura hipertextual en un bucle abierto. Para llevar a cabo esta operación basta con que presione el botón adecuado en la barra de herramientas y la herramienta enlazará, de manera interna (ya que el enlace no será representado gráficamente en el área de trabajo) el nodo de fin con el nodo de inicio.

Ya que no existe una representación gráfica del enlace interno entre el nodo final y el inicial, la herramienta varía el diseño del botón que abre o cierra la secuencia principal convirtiéndola en bucle abierto dependiendo del estado en el que se encuentre.

Esto es, si la secuencia lineal es tal y no ha sido convertida en un bucle abierto, el botón en la barra de herramientas tendrá el siguiente aspecto  . Si, por el contrario, usted ha decidido cerrar la secuencia principal, convirtiéndola de esta forma en un bucle abierto, el aspecto del botón de la barra de herramientas sería el siguiente  .

Nota sobre su uso

Convertir la secuencia principal (de forma lineal), en un bucle abierto, significa que la historia, a pesar de seguir teniendo un nodo que será el primero en ser presentado al lector cuando accede a ella, deja de tener un inicio y un final marcados.

El hecho de que cuando se ha llegado al final de la obra ésta te devuelva de nuevo al inicio, convirtiendo el relato de esta manera en algo cíclico, permite al lector no tener una sensación muy marcada de finalización de la obra, instándole a un mismo tiempo a su relectura siguiendo, en esta ocasión, nuevos caminos de lectura.

5. Funcionalidades avanzadas

Hasta este momento usted ha aprendido a utilizar todas aquellas funcionalidades que utilizará con más frecuencia durante la creación y edición de sus hiperficciones. En este apartado encontrará los detalles sobre aquellas funcionalidades que pueden encontrarse bajo el menú Herramienta -> Opciones y cuyo uso aún no ha sido explicado. Estas funciones tienen que ver con la personalización del funcionamiento de la herramienta y podrá modificar su configuración desde el cuadro de diálogo mostrado en la siguiente figura y que aparecerá al acceder a las mismas.

Figura 10. Opciones de personalización de la herramienta.

En este cuadro usted puede configurar las opciones de personalización mediante la selección de aquellas características que más se ajusten a sus deseos. Éstas serán aplicadas de forma inmediata a la herramienta en el momento en el que confirme los cambios pulsando el botón Aceptar.

5.1 Modo de visualización de etiquetas

Con el fin de facilitar la creación y el trabajo con diagramas de una complejidad considerable, y teniendo en cuenta que el espacio disponible en el área de dibujo es limitado, la herramienta ofrece tres modos distintos de visualización de las etiquetas asociadas a los enlaces existentes en la estructura hipertextual.

El primero de los modos es aquél en el que las etiquetas permanecen **siempre visibles.** Este es el modo por defecto, y en él las etiquetas que contienen la porción de texto asociada a cada enlace, o "<VACIO>" en el caso de que el enlace en cuestión aún no tenga ningún texto asociado, están siempre a la vista.

El segundo de los modos es aquél en el que las etiquetas permanecen **siempre ocultas.** El uso de este modo es recomendable cuando se está trabajando con diagramas de un tamaño y complejidad considerablemente grandes, ya que ayuda a "despejar" el área de trabajo de información de la que se puede prescindir durante la construcción de la red de hipertexto. Este modo es recomendable si usted ha adquirido ya una cierta experiencia en la escritura de hiperficción y es capaz de llevar a cabo su construcción de una forma ordenada.

Si por el contrario, usted no es del todo experto pero también necesita "despejar" el área de trabajo para poder trabajar de una forma más cómoda, el modo más recomendable es aquel denominado semi visible. En este modo, las etiquetas permanecen ocultas por defecto, pero la consulta de las mismas es fácil de llevar a cabo ya que basta con hacer **doble clic sobre el enlace** para que las etiquetas se hagan visibles. Si lo que desea es ocultar una etiqueta que con anterioridad hizo visible, simplemente vuelva a hacer doble clic sobre el enlace correspondiente y la etiqueta se ocultará.

5.2 Modo de navegación

En primer lugar es necesario explicar en qué consiste el modo de navegación. El editor de nodos presenta dos modos diferentes de trabajo, el modo edición (que se activa pulsando el botón ) y el modo navegación (que se activa pulsando en la barra de herramientas el botón ).

El modo de edición es el modo en el que se accede por defecto al editor, y en él se puede insertar texto e imágenes así como modificar los ya existentes. Se puede decir que es el modo de operación normal del editor.

El modo de navegación, por el contrario, desactiva todas las funciones de edición disponibles para permitir navegar por la red hipertextual creada. En el momento en el que este modo se activa, los enlaces contenidos en el texto del nodo y que aparecen marcados como tal pueden ser seguidos. Esto significa que usted puede hacer clic sobre ellos y el nodo destino del enlace se abrirá en una nueva pestaña en el editor.

Nota sobre su uso

El modo de navegación está especialmente pensado para que usted como autor pueda experimentar con la obra que está en proceso de creación de un modo similar al del lector cuando navega por ella.

Podrá seguir los enlaces para ir creando su propio camino de lectura, lo que le permitirá comprobar la coherencia del discurso o el relato.

Volviendo a la personalización de la herramienta, el modo de navegación del editor puede ser configurado de dos formas diferentes.

En la primera de ellas, la configuración por defecto, los nodos de destino que se abren al hacer clic sobre un determinado enlace se abren en una pestaña nueva que pasa a ser la pestaña activa en el editor y, por lo tanto, se muestra en primer plano.

La segunda de las configuraciones permite usar un panel auxiliar en el modo de navegación para mostrar los nodos destino de los enlaces. De esta manera, cuando haga clic sobre el enlace el área de trabajo de la herramienta se dividirá en dos mitades y a la derecha aparecerá un nuevo panel de pestañas. En este nuevo panel se abrirá el destino del enlace que usted acaba de activar.

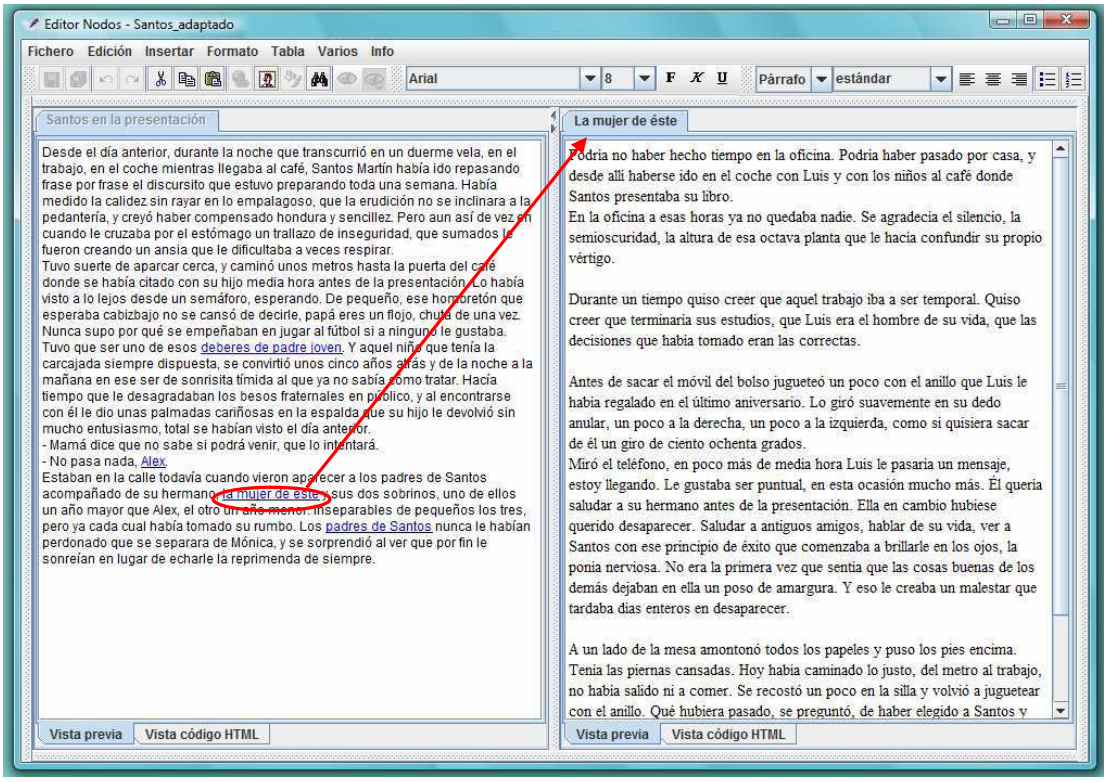


Figura 11. Editor de nodos en modo navegación usando el panel auxiliar.

Para activar el uso de este panel auxiliar usted debe marcar la casilla correspondiente a su activación que se muestra en la parte baja de la ventana de diálogo mostrada en la figura 10. Una vez usted haya activado esta opción, la apertura de nodos destino en el menú navegación seguirá las siguientes reglas:

- - Cuando se intenta abrir un nodo desde la herramienta, se comprueba si ese nodo ya está abierto en cualquiera de los dos paneles.

  - Si lo está, la pestaña en la que esté abierto pasa a primer plano y el panel que la contiene pasa a ser el activo.

- Si no lo está, el nodo se abre en el panel que se encontraba activo cuando se hizo doble clic en el nodo.

- Cuando se intenta abrir un nodo al activar un enlace con el modo de navegación activo, de nuevo se comprueba si ese nodo ya está abierto en cualquiera de los dos paneles.

  - Si lo está, la pestaña en la que esté abierto pasa a primer plano y el panel que la contiene pasa a ser el activo.

  - Si no lo está, el nodo se abre en el panel contrario a aquél en el que se produjo la activación del enlace, de manera que se ofrezca al autor la vista simultánea de origen y destino de un enlace.

HyperViewer
Manual de usuario

Contenido

1. Sobre HyperViewer

Como complemento a la herramienta de creación de hiperficción HyperAuthor, HyperViewer es una herramienta que permite visualizar el contenido de las obras creadas mediante el uso de la primera y que añade algunas características específicas que apoyan los principios en los que se fundamenta la literatura hipertextual.

Con un diseño muy sencillo, HyperViewer permite la navegación por los hipertextos de forma natural y cómoda para el usuario. Constituye, por lo tanto, un dispositivo lector que actúa de intermediario entre el lector y la obra, dotando además al usuario de herramientas de navegación que le ayuden en el proceso de lectura.

Al tratarse de un dispositivo lector, HyperViewer no ofrece funciones que le permitan modificar el contenido de los hipertextos y guardarlos en disco. La principal inquietud de esta herramienta consiste en proporcionar un entorno de lectura en el que se juegue con la estructura interna de la página y el propio hecho de pasar página para semejar el contenido presentado en pantalla con la página impresa.

HyperViewer es una aplicación especialmente diseñada para personas a quienes no se les presupone un gran conocimiento tecnológico, que quieren iniciarse en la lectura de literatura hipertextual. Pretende ser una herramienta interactiva, atrayente y divertida que le permita a usted, usuario, comprobar "in situ" cómo se puede leer hiperficción de una manera satisfactoria.

Para ayudarle en esta tarea, a lo largo de este manual encontrará algunas notas marcadas con el siguiente símbolo  . Estas notas le explicarán los conceptos y nociones básicas que no están directamente relacionado con HyperViewer sino con las estructuras o técnicas usadas en la lectura de literatura hipertextual y que, probablemente, le ayuden a fijar sus ideas. Si usted está familiarizado con este tipo de lectura, puede saltar estas notas sin miedo alguno.

Además de éstas, a lo largo del texto del manual encontrará otras notas marcadas con el símbolo  que le aclararán aspectos de tipo práctico (auxiliares) que puede que ya conozca, y que por lo tanto, sólo necesitará consultar si tiene una duda concreta.

2. Empezando

A estas alturas ya ha leído un poco sobre esta herramienta, lo que hace y sobre cómo y para qué debe usarse. Es, por lo tanto, un buen momento para echarle un vistazo general. Para empezar con buen pie deberá, por supuesto, iniciar la aplicación.

Iniciando HyperViewer

HyperViewer se proporciona en dos formatos diferentes, cada uno de los cuales tiene su propia forma de inicialización:

- *Ejecutable (.exe). Solo para sistemas Windows. Para iniciar la aplicación basta con hacer doble clic sobre el archivo .exe*

- *Archivo jar. Válido tanto en sistemas Windows como Linux. La aplicación puede iniciarse de dos formas:*

  a) *En línea de comandos mediante la instrucción "java –jar HyperViewer.java".*

  b) *Haciendo doble clic sobre el archivo .jar si dispone de una versión de java mayor o igual a 1.6*
  (usted puede comprobar este punto tecleando "java –version" en su línea de comandos).

En la siguiente figura se muestra la vista principal de la aplicación. Esta ventana aparecerá en su pantalla tras indicar, en el cuadro de diálogo previo, el archivo que contiene la hiperficción que desea reproducir.

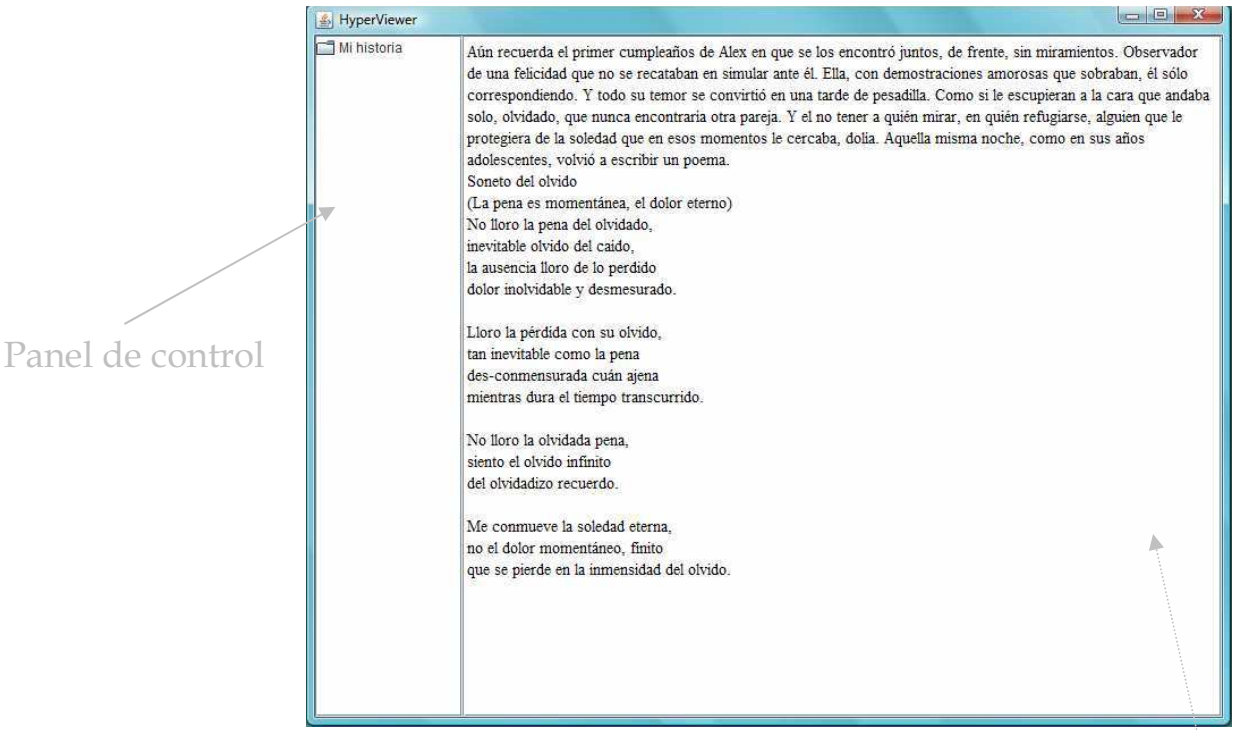


Figura 1. Vista de la interfaz de HyperViewer.

Como se puede observar con un simple vistazo, HyperViewer, a diferencia de la gran mayoría de las aplicaciones actuales, no dispone de una barra de menús al uso, así como tampoco dispone de una barra de herramientas que contenga las principales funciones de la aplicación.

Todas aquellas acciones de usuario relacionadas con la lectura del hipertexto se encuentran "integradas" dentro del propio área de lectura, de forma que ésta responderá de forma diferente a sus clic de ratón dependiendo de la zona en la éstos se produzcan.

De manera adicional, y con el fin de maximizar el espacio dedicado al área de lectura dentro de la ventana de la aplicación, en la zona izquierda de la misma aparece lo que se denomina panel de control. El contenido de este panel consiste básicamente en todas aquellas funciones distintas a las asociadas estrictamente con la navegación.

En esta versión de HyperViewer la única funcionalidad incluida en el panel de control es el histórico de navegación. Este histórico se encuentra bajo el nombre "Mi historia".

Histórico de navegación

Uno de los principales inconvenientes que experimentan los lectores durante su navegación por los hipertextos es la desorientación. La constante toma de decisiones de navegación pueden provocar la pérdida de la referencia "espacial" y el lector con frecuencia se encuentra perdido en la red hipertextual.

Las ayudas a la navegación son características que se incorporan a las herramientas de lectura y que están diseñadas para ayudar a que este fenómeno de desorientación no ocurra. El histórico de navegación se encuentra entre estas ayudas a la navegación, y consiste en una lista ordenada que contiene todos aquellos nodos por los que el lector ha pasado. De esta manera el lector puede reproducir su propio camino de lectura dentro de una determinada hiperficción si así lo desea y tiene además, acceso directo a cualquiera de los nodos ya visitados.

Para añadir cada uno de los nodos por los que usted ha pasado al histórico de navegación no es necesario realizar ninguna acción concreta, ya que este histórico es creado de forma automática por la herramienta.

Aviso

Los nodos en el histórico de navegación permanecerán ocultos por defecto. Para poder tener acceso a este histórico despliegue la lista de nodos haciendo doble clic con el ratón sobre la carpeta con el nombre "Mi historia".

En el centro de la ventana, ocupando el espacio mayor, se encuentra el área de lectura donde, como ya se ha comentado, se muestra el contenido de los nodos que componen la obra que está leyendo. Este área de lectura posee además la "inteligencia" necesaria para entender sus clics de ratón y permitir, por lo tanto, la navegación por el hipertexto sin la necesidad de que usted desvíe su atención del relato.

Hasta este momento se ha realizado un breve recorrido a vista de pájaro por la aplicación. Este recorrido le habrá resultado de ayuda para entrar en contacto con la herramienta; aún así, sería buena idea que antes de comenzar la lectura de los siguientes apartados se familiarizase un poco con ella explorando las opciones ofrecidas por el área de lectura así como por su histórico de navegación.

2.1 Abrir una hiperficción

Por supuesto, la apertura de un hipertexto es la primera y más fundamental de las operaciones a realizar con HyperViewer, ya que el objetivo de esta aplicación no es otro que el que usted pueda visualizar y navegar por hipertextos para llevar a cabo su lectura.

Comenzaremos por la pantalla que aparece nada más iniciar la aplicación. Esta pantalla, que se muestra en la siguiente figura, le insta a que introduzca el nombre del archivo que contiene el hipertexto cuya lectura quiere comenzar.

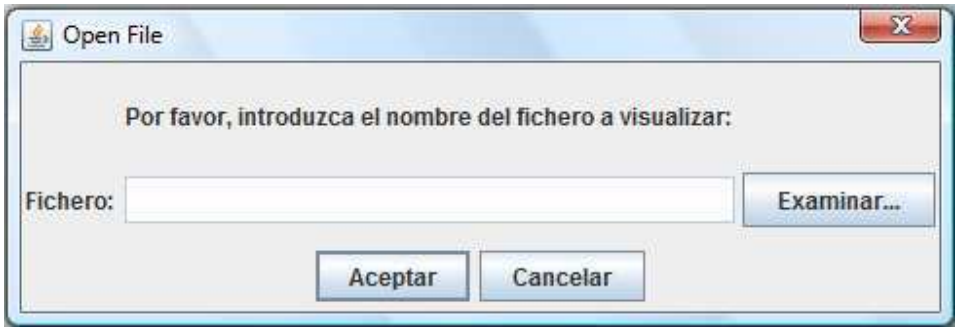


Figura 2. Ventana inicial donde indicar a la aplicación el archivo a visualizar.

Pulse sobre el botón Examinar para navegar por las carpetas de su sistema y localizar el archivo deseado. Una vez el nombre aparezca en el campo de texto, pulse Aceptar. En el área de lectura aparecerá el contenido del nodo inicial de la hiperficción que usted está tratando de visualizar. A partir de este momento debe utilizar los mecanismos de navegación ofrecidos por la aplicación para moverse libremente por el hipertexto. Para una explicación detallada de los mecanismos de navegación soportados por HyperViewer, por favor, consulte el apartado 3 de este manual.

2.2 Enlaces en HyperViewer

Enlaces

Un enlace no es más que la representación de un camino de bifurcación dentro de la estructura hipertextual. De manera general, los enlaces se representan como una porción de texto marcada de manera especial dentro de un determinado documento. Los enlaces se caracterizan por actuar como nexo de unión entre un documento origen y otro destino (o dos fragmentos distintos dentro del mismo documento).

Se puede pensar en un enlace como un puntero a un lugar distinto al que se encuentra. De manera habitual, origen y destino contienen información relacionada. De esta manera el usuario puede acceder a la información del destino en el momento en el que lo desee.

Cada enlace está representado por una porción de texto, normalmente denominada nombre, con la cual se le reconoce.

Los enlaces pueden ser de distinto tipo dependiendo del tipo de relación que establezca entre origen y destino. Puede haber enlaces que creen jerarquías o enlaces que comuniquen información a un mismo nivel. También existen enlaces interno o externos dependiendo de si el destino está fuera o dentro del documento en el que está contenido, etc.

En los hipertextos que puede leer en HyperViewer se utilizan dos tipos diferentes de enlaces y, por lo tanto, también dos tipos diferentes de navegación. Estos son los enlaces secuenciales y los enlaces de bifurcación.

El primer tipo de enlace debe su nombre al hecho de que permite que el flujo de lectura se dé en los dos posibles sentidos. Este tipo de enlaces le permite a usted como lector una navegación del tipo anterior o siguiente similar a la de los libros tradicionales. Note que estos enlaces no tienen ningún texto asociado en el contenido del nodo que se representa en pantalla, por lo que son de alguna manera "invisibles" al lector.

Los enlaces de bifurcación, por su parte, representan la "puerta de entrada" a un nuevo camino de lectura. Este segundo tipo de enlaces es, por lo tanto, aquel que implica una ruptura en la linealidad de la lectura. Si la obra que está visualizando no se reduce a una simple estructura lineal es necesario que utilice este último tipo de enlaces, ya que serán los que le permitan sacar el máximo partido de la literatura hipertextual. Estos enlaces sí que tienen un texto asociado en el contenido del nodo y para seguir el camino de lectura al que dan lugar, basta con hacer clic sobre ellos.

3. Navegando por una hiperficción

Una vez usted tiene en pantalla el contenido del primero de los nodos de la hiperficción que desea leer, es el momento de comenzar la navegación por ella. En este apartado se exponen los detalles del uso de las diferentes opciones que HyperViewer le ofrece para navegar por un hipertexto.

Formato de las hiperficciones

Todo hipertexto abierto en HyperViewer debe contener una estructura principal de forma lineal. Esta estructura llevará el peso del relato y será la encargada de hacer avanzar la acción. La navegación que se da en este tipo de secuencias es la de anterior y siguiente.

La ruptura de esta linealidad se da con otro tipo de estructuras que ramifican a partir de las secuencias lineales. Para acceder a estas estructuras se debe activar un enlace de bifurcación usando la navegación mediante enlaces. Una vez se haya entrado en un nuevo camino de lectura, el tipo de navegación vuelve a ser del tipo anterior o siguiente.

Estos dos tipos diferentes y diferenciados de navegación, asociados a los diferentes tipos de enlaces, ayudan al lector a "orientarse" dentro del hipertexto, a saber con mayor exactitud su posición dentro de la red hipertextual y sobre todo, a hacerse una idea sobre qué es lo que viene a continuación de manera que el hecho de tomar una bifurcación en el relato no resulte desconcertante.

La primera noción que debe tener clara es que, como ya se ha comentado con anterioridad, todas las funcionalidades relacionadas con la navegación por el hipertexto están integradas en el área de lectura. Ésta se encuentra "dividida" (aunque no de una forma visible) en tres zonas diferenciadas, cada una con un tamaño igual a un tercio del tamaño del área de lectura completa y situadas de manera que ocupen las zonas izquierda central y derecha de la misma.

Si usted hace clic sobre cada una de estas zonas obtendrá, por parte de HyperViewer, una respuesta diferente en cada caso. Expongamos esto en detalle:

3.1 Navegación anterior-siguiente

Este tipo de navegación utiliza las franjas izquierda y derecha de la pantalla de la siguiente manera: si usted hace clic sobre la franja izquierda de la pantalla, el lector vuelve al nodo anterior al que estaba visualizando; de una manera similar, al hacer clic sobre la franja derecha, usted avanza en el relato haciendo aparecer el nodo siguiente en pantalla.

Estas dos zonas están activas permanentemente y en cualquier nodo. Esto quiere decir que este tipo de navegación estará siempre disponible independientemente de que el nodo presentado en pantalla contenga o no enlaces de bifurcación (aquellos que aparecerían marcados de manera especial en el texto).

La excepción a esta regla se encuentra en el nodo inicial y quizá también, en el final. En el nodo inicial no se podrá activar la zona izquierda para llegar al nodo anterior, ya que ese nodo no existe. De manera similar, en el nodo final puede que no sea posible activar la franja derecha para llegar al nodo siguiente ya que, de nuevo, ese enlace puede no existir.

Final de un hipertexto

En el caso del nodo final, el nodo siguiente puede existir o no dependiendo de si la secuencia principal del relato tiene forma lineal o forma de bucle abierto. Si es lineal, el final es único y no existirá nodo siguiente; en caso contrario, el nodo siguiente al nodo "final" será de nuevo el nodo inicial.

Cuando usted se mueve por el hipertexto utilizando estas franjas, sabrá que está navegando por una secuencia lineal; el relato avanza hacia su final del modo acostumbrado en la literatura tradicional.

3.2 Navegación usando enlaces

El tipo de navegación explicada en detalle en el apartado anterior no es suficiente para lograr una lectura satisfactoria de hiperficción. Las obras hipertextuales no están formadas únicamente por estructuras lineales en las que el lector se puede mover usando anterior y siguiente, existen también otro tipo de estructuras que abren nuevos caminos de lectura rompiendo la linealidad de la obra.

Para acceder a estos nuevos caminos es indispensable el uso de la navegación mediante enlaces. Este tipo de navegación utiliza la franja central del área de lectura, de la que aún no hemos hablado.

Estructuras no lineales

Este tipo de estructuras, si bien implican una ruptura de la linealidad en la secuencia de la que provienen, son en sí mismas secuencias lineales con origen y final en un mismo nodo, y el usuario se mueve dentro de ellas como si de una secuencia lineal al uso se tratara, es decir, usando la navegación de tipo anterior-siguiente.

Cuando un nodo es presentado en pantalla por la aplicación, la primera tarea a realizar por usted como usuario lector es, obviamente, proceder a su lectura. Si así lo ha hecho, se habrá dado cuenta de que los posibles enlaces contenidos en el texto del nodo permanecen "ocultos", lo que quiere decir que no aparecen marcados de una manera especial y diferente a la del resto del texto del nodo. Véase un ejemplo en la siguiente figura:

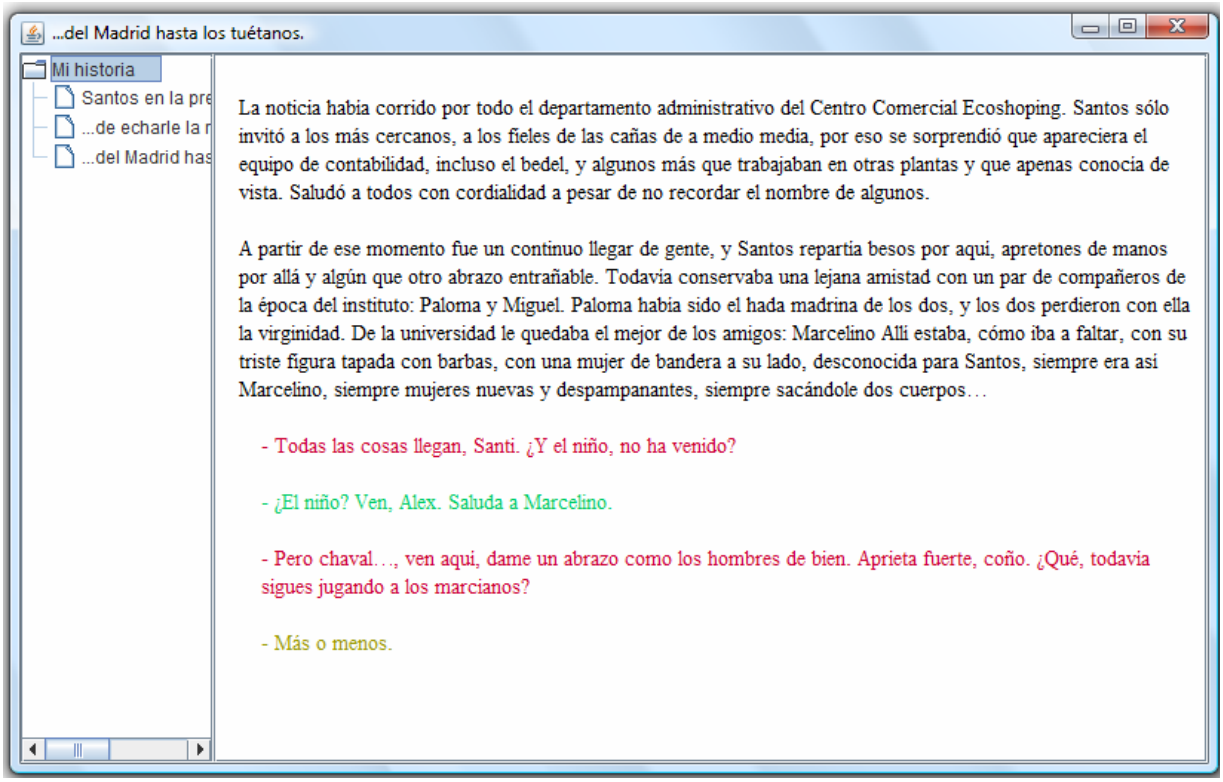


Figura 3. Vista de un nodo en HyperViewer tras acceder a él.

Valor de los enlaces

En HyperViewer, al lector no le está permitida la toma de decisiones de navegación hasta que no ha completado la lectura del nodo. Poder saltar y tomar un nuevo camino de lectura en cualquier momento constituye un grado de libertad que compromete la

creación de la obra. El texto del nodo junto con los enlaces y las imágenes que contiene conforman una única unidad de significado. Por lo tanto, para que el lector tenga la oportunidad de descubrir algo de sí mismo atendiendo a lo que el autor diseñó, esta unidad de significado debe ser asimilada de forma completa por el lector antes de que la herramienta le permita seguir los enlaces que contiene el nodo.

Con esta finalidad, cuando se accede a un determinado nodo, la herramienta no revela desde el primer momento qué palabras de las contenidas en el texto del nodo son realmente enlaces. Éstos aparecen imbricados en el texto pero no implícitamente marcados, por lo que se consigue que todas las palabras tengan el mismo valor para el lector cuando éste las lee.

Sin embargo, si usted desea tomar una nueva ruta, una que se salga de la linealidad de la secuencia en la que se encuentra para poder seguir la pista de un concepto, una idea, un personaje, un lugar o una historia que le interese en ese momento, entonces debe utilizar la activación de los enlaces que aparecen en el texto.

Pero, ¿cómo activar los enlaces si aún no sabe cuáles son? La respuesta es sencilla. Cuando haya terminado la lectura del nodo, haga clic sobre la franja central del área de lectura y verá como se "revelan" los enlaces contenidos en el texto. Las palabras que actúan como tal se marcarán en letra azul y subrayada. En la siguiente figura se muestra el mismo nodo que en la figura anterior. En esta ocasión los enlaces ya han sido hechos visibles.

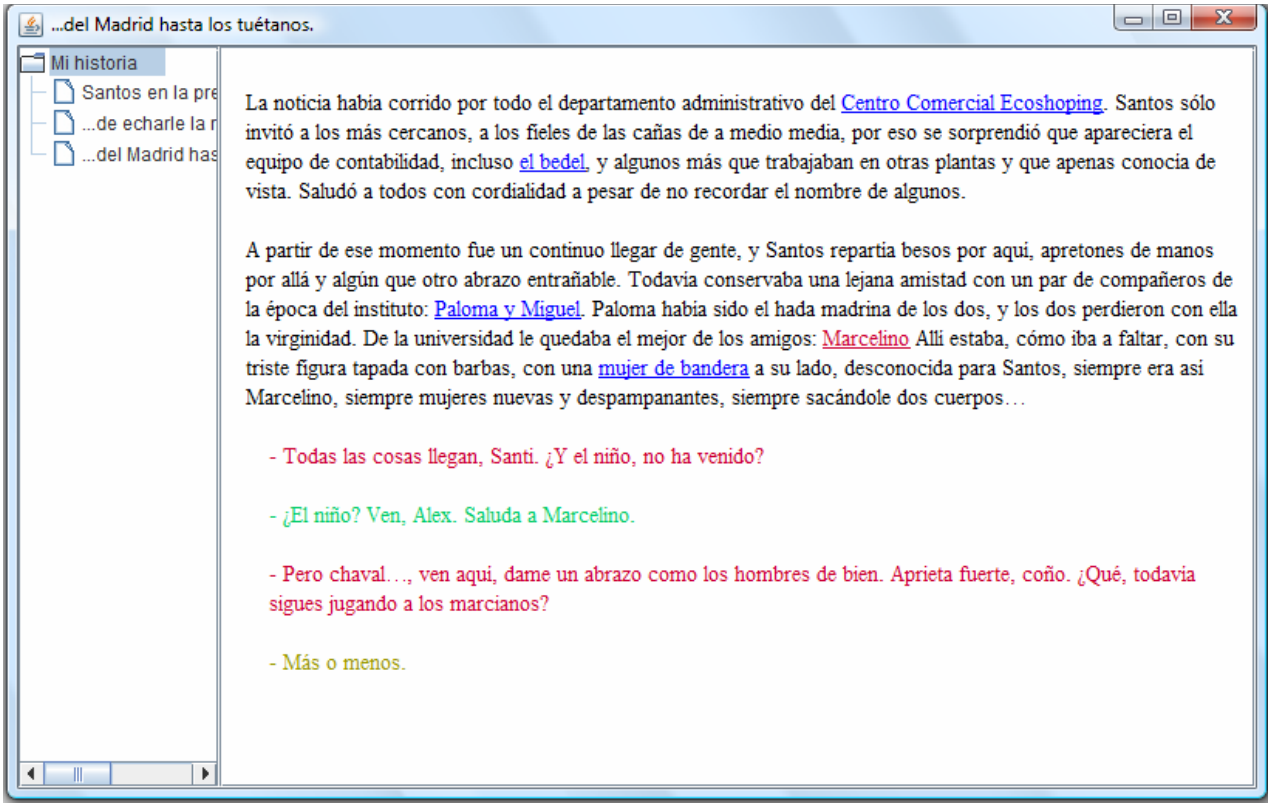


Figura 4. Vista de un nodo en HyperViewer tras activar sus enlaces.

Al pulsar sobre una de las palabras marcadas como enlaces, el visor abre y presenta en pantalla el contenido del nodo destino del enlace; para usted, lector, queda abierto, en ese momento, un nuevo camino de lectura.

Si por el contrario, el texto que usted está visualizando no contiene ningún enlace de bifurcación, no piense que ha llegado a un punto muerto en el relato. Lejos de eso, lo que debe entender es que se encuentra en un nodo perteneciente a una secuencia lineal y que, por lo tanto, es posible usar las franjas anterior y siguiente, que permanecen siempre activas, para desplazarse por ellas.

Para ayudarle a recordar sus opciones de navegación, cuando se encuentre en la situación de que está visualizando un nodo que no contiene enlaces y ha hecho clic en la zona central, la herramienta en este caso, en vez de mostrar los enlaces como en ocasiones anteriores, lo que hace es resaltar, de nuevo usando el azul, las dos zonas que usted tiene disponibles para continuar su lectura.

Puede ver un ejemplo de este hecho en la siguiente figura:

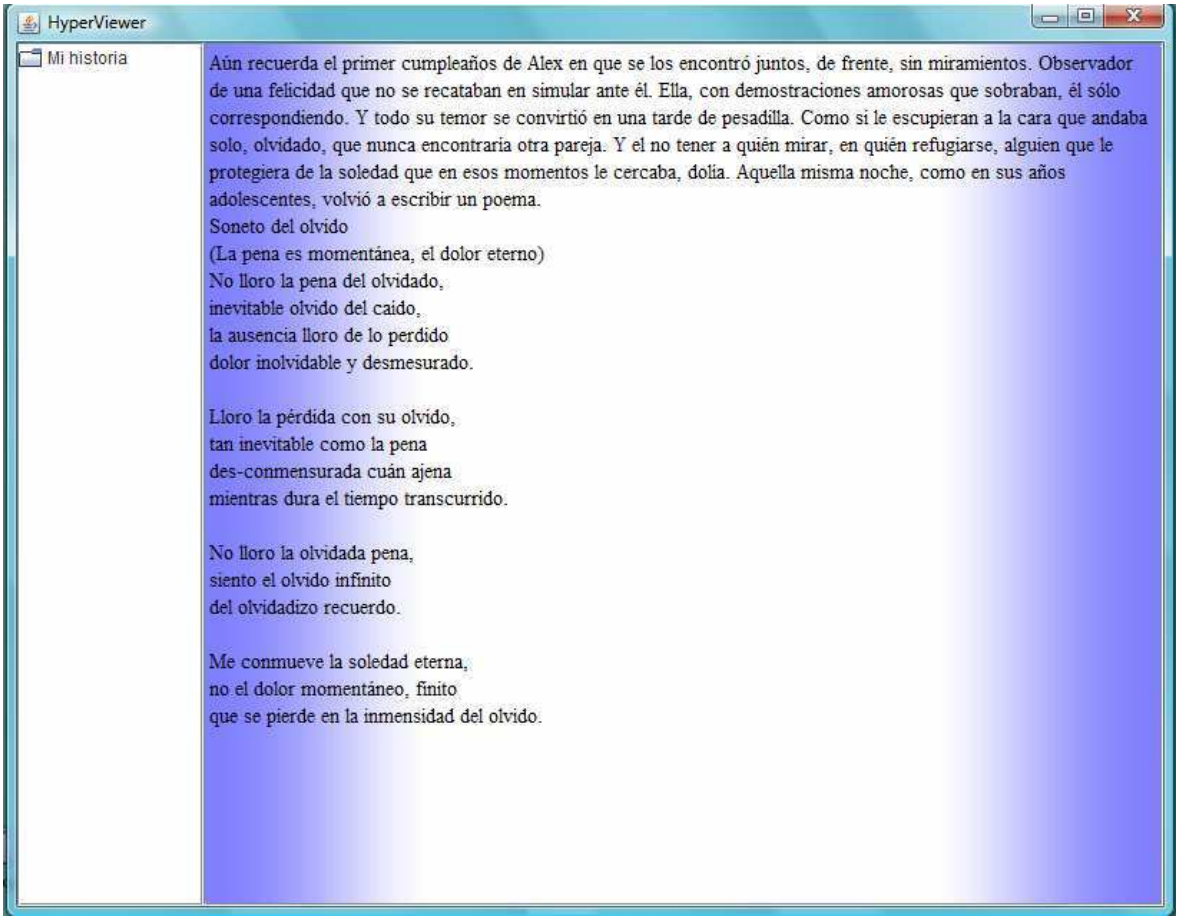


Figura 5. Vista de las franjas que aparecen en pantalla al tratar de activar los enlaces de un nodo que no contiene ninguno.

Pliego IV
Presupuesto

Presupuesto

El presupuesto para la realización de este proyecto está constituido por los siguientes conceptos:

1. Costes por tiempo

Los costes de ejecución que se establecen en este proyecto como coste por tiempo se corresponden con las tablas salariales del *Convenio Colectivo Nacional de empresas de ingeniería y oficinas de estudios técnicos* y las últimas tablas para el año 2009 publicadas en el BOE por el Ministerio de Trabajo e Inmigración:
[http://www.boe.es/boe/dias/2009/03/28/pdfs/BOE-A-2009-5211.pdf#]

- ⇒ Coste salarial anual para Nivel 1 (Licenciados y Titulados) : 22937,51€
- ⇒ Coste hora normal (asumiendo 1806 horas anuales) : 12.7 €

Labor	Días Totales	Horas/día	Horas totales	Importe
Estudios iniciales	50	4	200	2.540 €
Diseño de la aplicación	80	4	320	4.064 €
Implementación	150	4	600	7.620 €
Documentación	70	4	280	3.556 €
TOTAL				**17.780 €**

Tabla 1. Desglose de costes de tiempo

2. Costes materiales

2.1. Software

- ⇒ **Software propietario**. Se incluyen en esta categoría el sistema operativo y las herramientas de desarrollo utilizadas.

Licencias	Coste total	Coste asociado
Windosws XP Professional	300 €	30 €
Office Professional 2003	800 €	80 €
TOTAL		**110 €**

Tabla 2. Desglose de costes de software propietario

- ⇒ **Software de libre distribución**. En esta categoría se encuentran todas aquellas herramientas empleadas cuyo uso no conlleva ningún tipo de coste asociado. En este caso han sido Eclipse, Install4j, ObjectAid UML Explorer y Gimp.

2.2. Hardware

En esta sección se incluyen los costes derivados del empleo de equipos.

Concepto	Coste total	Coste asociado
Equipo Pentium IV 2.8 GHz	2500 €	250 €
TOTAL		**250 €**

Tabla 3. Desglose de costes de hardware

2.3. Otros

En esta sección se incluyen todos aquellos costes materiales que no se pueden encuadrar en ninguno de los apartados anteriores.

Concepto	Coste total	Coste asociado
Conexión Internet ADSL	400 €	40 €
TOTAL		**40 €**

Tabla 4. Desglose de costes materiales encuadrados en "Otros"

3. Presupuesto total

Considerando todos los recursos humanos empleados contabilizados como costes en tiempo así como los materiales empleados contabilizados como recursos materiales, el presupuesto necesario para la realización de este proyecto se corresponde con el valor mostrado en la siguiente tabla.

Licencias	Importe total
Costes en tiempo	17.780 €
Costes materiales	400 €
TOTAL	**18.180 €**

Tabla 5. Presupuesto total